(2)

Report DARPA/Final/0490

OPTICAL NEURAL NETS FOR SCENE ANALYSIS

David Casasent (PI)
Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213
(412) 268-2464

March 1990

Final Report for First Year: April 1989 - March 1990
Contract DAAH01-89-C-0418
31 March 1989 - 31 March 1992

UNCLASSIFIED - DISTRIBUTION UNLIMITED

Sponsored by:
Defense Advanced Research Projects Agency (DoD)
Defense Sciences Office
ARPA Order Nr. 6671
Issued by U.S. Army Missile Command
Contract #DAAH01-89-C-0418

DISCLAIMER

The Contractor, Carnegie Mellon University, hereby certifies that to the best of its knowledge and belief, the technical data delivered herein on Contract DAAH01-89-C-0418 is complete, accurate, and complies with all requirements of the contract.

"The view and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government."

Date 30 March '90

D. Casasent(PI) David Casasent

FINAL YEAR 1 REPORT
Contract DAAH01-89-C-0418
31 March 1989 - 31 March 1992
Period Covered: April 1989 - March 1990
Date: March 1990

OPTICAL NEURAL NETS FOR SCENE ANALYSIS

ABSTRACT

Our objective is to develop new neural net algorithms and architectures for scene analysis and to demonstrate them on a fabricated new hardware laboratory neural net. Our approach marries pattern recognition and neural net techniques and optical/digital technologies. Our hardware laboratory system uses digital and optical neural net hardware in an analog neural net. Our algorithms are intended to be useful on such low accuracy analog hardware. Our algorithms cover a wide range of neural net algorithms and architectures. These can all be utilized on the same laboratory hardware. Our algorithms include five new optimization neural nets (matrix-inversion, mixture, multitarget tracking, symbolic, and production system neural nets) and an adaptive neural net (adaptive clustering neural net).

1. INTRODUCTION

1.1 OBJECTIVE

Our objective is to produce new neural net algorithms and architectures for use in scene analysis. We also intend to demonstrate these algorithms in real-time on a new hybrid laboratory system. We consider a number of optimization neural nets and one adaptive neural net. A unique aspect of our work is that all of these neural net algorithms can be implemented in real-time on the same multifunctional neural net hardware laboratory system. Another unique aspect of our work is its attention to marrying both pattern recognition and neural net techniques. We view a major property of neural nets to be their ability to produce nonlinear decision surfaces (as are needed for difficult pattern recognition problems - those for which neural nets are required).

1.2 APPROACH

Our approach is hybrid and multidisciplinary. We marry pattern recognition and neural net techniques. We also marry optical and digital technologies. A hybrid neural net thus results. We also concentrate on the use of one basic hybrid architecture that is useful for implementing various optimization and adaptive neural nets. Our work thus distinguishes between these two general classes of neural nets (optimization and adaptive) with both being realizable on the same basic hybrid architecture.

1.3 OVERVIEW

Chapter 2 provides an overview of our processor with attention to its multifunctional nature and its general architecture. Chapter 3 details the present status of the system, and our present simulation status of it for one specific neural net (mixture neural net). This is the first meaningful neural net simulation.

Chapters 4-9 then detail our five specific optimization neural nets. Chapter 4 presents our mixture neural net (applied to an imaging spectrometer case study). Chapters 5 and 6 present our multitarget tracking neural net work with attention to a cubic energy neural net (Chapter 5) and a preferable quadratic energy neural net (Chapter 6) requiring a simpler optical processor. Chapters 7-9 summarize simulated (Chapter 7) and laboratory (Chapter 8) data on our symbolic and production system neural nets based on our initial concepts (Chapter 9).

Our adaptive neural net research is included in our extensive summary of problems in present neural nets and a new adaptive clustering neural net using pattern recognition and neural net techniques (Chapter 10).

Section 1.4 provides a summary of the various neural nets we have considered. Papers published and submitted during the first year of this project follow in Section 1.5. These 10 papers represent an enormous one year output and indicate the completeness with which we have treated all aspects of neural nets for scene analysis with attention to new algorithms and applications, a combination of optical/digital techniques for implementation, a multifunctional hybrid optical/digital neural net architecture, its laboratory realization and a new adaptive clustering neural net algorithm (combining pattern recognition and neural net techniques). This effort is thus quite significant in terms of algorithms, architectures, and hardware.

1.4 SUMMARY OF NEURAL NETS (NNs) CONSIDERED

The seven neural nets we have considered are now briefly summarized.

The input neurons to the production system NN are facts (antecedents and consequents). Objects and object parts are used in our initial work. Surface types for object parts (cylinder, sphere, valley, ridge, etc.) can also be used in future work. The objects are typical of those present in various scenes. The weights define the rules. These are initially posed as if-then statements, with all rules written as the AND of several antecedents and the OR of several such sets of antecedents. The output neurons that fire represent the new facts that are now learned to be true. As the system iterates, it learns new rules and infers new results on the present input data. We initially consider a propositional calculus system (with all parameters being exact terms) and then plan to address a predicate calculus system (with parameters being variables) that is much more powerful.

The E input neurons in our mixture NN each correspond to the fractional amounts of E elements present in a mixture of elements within one region of an input scene. The outputs from two matrix-vector multiplications are combined to form the new neuron states. After a number of iterations, the final neuron states denote the fractional amount of each element present in the input mixture.

The matrix inversion NN produces the inverse of a matrix that is given to the processor. To calculate the inverse $\underline{X}$ of a matrix $\underline{Q}$, we realize that $\underline{Q}\underline{X} = \underline{I}$. We formulate the solution (the elements of the inverse of Q) as the minimization of an energy function . We solve for the X that minimizes the energy function on a neural net. The matrix elements (weights) in this NN have an attractive block Toeplitz form and thus acousto-optic (AO) architectures should be very suitable for implementing this NN. This represents the first AO NN. Since matrix inversions are required in many pattern recognition linear discriminant function designs and in most adaptive algorithms, this NN should have general computational use in image processing (as well as in adaptive radar, control, etc.).

The cubic energy NN for MTT takes measurements on objects in each of three frames and it assigns one target per measurement and time frame. This is useful for time sequential scene analysis to associate objects (or object parts) in several time frames.

The quadratic energy MTT NN is a simplified version of the cubic energy NN. It processes pairs of time frames. The resultant optical architecture is much simpler than the cubic energy NN and significantly reduces component requirements.

The symbolic NN combines a symbolic correlator, production system NN, feature extractor and image processing NN. Its major advantage is the ability to process multiple objects in the field of view (this is achieved by the symbolic correlator). No other NN has this ability. It outputs a symbolic description of each region of the input that denotes which generic shapes are present and their location. These data are then symbolically encoded and fed to an NN. The NN is unique because of its symbolic input neuron representation. Alternatively, the locations of regions of interest in the input scene are used to guide the positioning of window functions (for segmentation) from which input features are extracted and subsequently fed to an NN for object classification. These NNs again combine pattern recognition and NN techniques.

The adaptive clustering NNis our major effort. The input neurons are features, the hidden layer neurons are prototypes of the various classes of objects and the output neurons denote the class of the input object. Clustering techniques are used to select the original hidden layer neurons (we allow several neurons or clusters per object class) and hence the initial input to hidden layer weights. These represent a set of linear discriminant functions (LDFs). The output neurons define the class of the input. The hidden to output layer weights map the clusters to classes. Our study of criterion functions determined the type of error function used to train the NN. Thus, advanced pattern recognition techniques are used to initialize the set of NN weights. A new adaptive NN learning algorithm is then used to refine and improve the initial weight estimates and to produce the LDF combinations that provide the nonlinear piecewise discriminant surfaces finally used. This is the adaptive learning stage. This new NN combines pattern recognition and NN techniques.

1.5 REFERENCES (DARPA Publications on this Year 1 Effort)

REFERENCES

1. E. Barnard and D. Casasent, "Image Processing for Image Understanding with Neural Nets", *Proc. IEEE International Joint Conference on Neural Networks (IJCNN)*, June 1989, Washington, D.C., Vol. 1, pp. I-111-115.

2. D. Casasent and E. Botha, "A Symbolic Neural Net Production System: Obstacle Avoidance, Navigation, Shift-Invariance and Multiple Objects", Proc. SPIE, Vol. 1192, November 1989.

3. E. Barnard and D. Casasent, "An Optical Neural Net for Classifying Image-Spectrometer Data", Applied Optics, Vol. 28, pp. 3129-3133, 1 August 1989.

4. D. Casasent and T. Slagle, "A Hybrid Optical/Digital Neural Network", Proc. SPIE, Vol. 1215, January 1990.

5. E. Barnard and D. Casasent, "Optical Neural Net for Matrix Inversion", Applied Optics, Vol. 28, pp. 2499-2504, 1 July 1989.

6. E. Barnard and D. Casasent, "Multitarget Tracking with Cubic Energy Optical Neural Nets", Applied Optics, Vol. 28, pp. 791-798, 15 February 1989.

7. M. Yee, E. Barnard and D. Casasent, "Multitarget Tracking with an Optical Neural Net Using a Quadratic Energy Function", Proc. SPIE, Vol. 1192, November 1989.

8. E. Barnard and D. Casasent, "A Comparison between Criterion Functions for Linear Classifiers, with an Application to Neural Nets", IEEE Trans. Systems, Man and Cybernetics, Vol. 19, No. 5, pp. 1030-1041, September/October 1989.

9. D. Casasent and E. Barnard, "An Adaptive-Clustering Optical Neural Net", Applied Optics, submitted August 1989.

10. D. Casasent, "A Multi-Functional Hybrid Optical/Digital Neural Net", Proc. SPIE, Vol. 1294, April 1990.

CHAPTER 2

"A Multi-Functional Hybrid Optical/Digital Neural Net"

PROC. SPIE, Volume 1294
April 1990, Orlando, Florida

A MULTI-FUNCTIONAL HYBRID OPTICAL/DIGITAL NEURAL NET

David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

A multi-functional hybrid neural net is described. It is hybrid since it uses a digital hardware Hecht-Nielsen Corporation (HNC) neural net for adaptive learning and an optical neural net for on-line processing/classification. It is also hybrid in its combination of pattern recognition and neural net techniques. The system is multi-functional. It can function as an optimization and adaptive pattern recognition neural net, as well as an auto and heteroassociative processor.

1. INTRODUCTION

Neural nets (NNs) have recently received enormous attention [1-2] with increasing attention to the use of optical processors and a variety of new learning algorithms. Section 2 describes our hybrid NN with attention to its fabrication and the role for optical and digital processors. Section 3 details its use as an associative processor. Section 4 highlights is use in 3 optimization NN problems (a mixture NN, a multitarget tracker (MTT) NN, and a matrix inversion NN). Section 5 briefly notes it use as a production NN system and symbolic NN. Section 6 describes its use as an adaptive pattern recognition (PR) NN (that marries PR and NN techniques).

2. HYBRID ARCHITECTURE

Figure 1 shows our basic hybrid NN [3]. The optical portion of the system is a matrix-vector (M-V) processor whose vector output P_3 is the product of the vector at P_1 and the matrix at P_2. An HNC digital hardware NN is used during learning to determine the interconnection weights for P_2. If P_2 is a spatial light modulator (SLM), its contents can be updated (using gated learning) from the digital NN. The operations in most adaptive PR NN learning algorithms are sufficiently complex that they are best implemented digitally. In addition, the learning operations required are often not well suited for optical realization; for optimization NNs, the weights are fixed; and in adaptive learning, learning is off-line and once completed the weights can often be fixed.

Four gates are shown that determine the final output or the new P_1 input neurons (Depending on the application). We briefly discuss these cases now and detail how each arises in subsequent sections. In most optimization NNs, an external vector $\underline{a}$ is added to the P_3 output (Gate 1 achieves this). In all NNs, a nonlinear thresholding (P_3 outputs are 0 or 1), truncation (all P_3 outputs lie between 0 and 1), or maximum selection (the maximum P_3 output is set to 1 and all other P_3 outputs to 0)

operation is required (Gate 2 achieves this). In a 3-layer NN (adaptive PR), a second M-V operation is performed with the P_3 neurons as inputs (Gate 3 achieves this). When the neuron outputs are fed back to the P_1 input neurons (in an iterative NN), linear or binary P_1 neurons may be used (Gate 4 produces the proper P_1 input neurons).

Figure 2 shows the optical portion of the NN as fabricated. A stripe electroded liquid crystal device (LCD) serves as P_1 and a computer-generated hologram (CGH) as the P_2 set of interconnection weights [4] (this allows real-time realization on an SLM with minimum space bandwidth product and/or improved light efficiency to avoid the large insertion loss of a standard M-V processor). An IBM PC/AT, DT 2821 data acquisition board, and various special-purpose hardware provide all digital/electronic support functions.

3. ASSOCIATIVE PROCESSOR (AP) USES

The first type of NN we consider is an AP. We consider pseudoinverse [6] and other advanced APs (such as the Ho-Kashyap AP) [5], since they have more storage capacity and better noise performance than Hopfield and other APs. They also require only one pass through the P_1-P_3 system (rather than many iterations, as in other APs). Thus, these systems use only the P_1-P_3 M-V processor (the P_1 input is the key vector, the matrix at P_2 is fixed and the P_3 output is the recollection vector - most closely associated with the input key vector). We emphasize heteroassociative processors (HAPs), since they make decisions (i.e. the recollection vector encoding denotes the class of the input key vector data). We also employ P_1 input neuron spaces that are features, facts, or symbolic descriptions of an object (this significantly reduces the dimensionality required - the number of neurons). With only 32 input neurons, we have recognized over 1000 distorted input objects in 10 classes. For these neural nets, the P_1 neurons are linear, the P_2 weights are analog, and the P_3 neurons are binary (or use maximum selection). We note that P_2 can also be the data matrix, in which case a nearest neighbor AP NN results.

NEURON REPRESENTATION SPACE

A key issue in all of our NN AP systems has been the use of a variety of neuron representation spaces. These include: iconic (one neuron per pixel in an image), feature spaces, symbolic (facts etc.) data, and encoded correlator output data. Iconic neuron spaces require a very large number of neurons and are thus not attractive. They also result in neuron spaces that are not distortion or shift invariant (i.e. they require many training images, one for each possible shift or distortion). Feature space representations result in fewer neurons (a considerable reduction in dimensionality) with various levels of distortion and/or shift invariance and are thus very attractive and preferable. Symbolic and correlator representations allow multiple objects to be handled (all other neural systems require preprocessing to isolate one object in the field of view, before inputing the data to a NN for processing).

4. OPTIMIZATION NEURAL NETS

A major class of NNs are optimization NNs (rather than PR NNs). In most such cases, these NNs are characterized by iterative M-V processors with a fixed P_2 set of interconnection weights, the addition of an external vector $\underline{a}$ to the P_3 output, and a nonlinear function (before the output is fed

back to P_1). The fixed P_2 weights allow us to efficiently use the optical portion of Figure 1 with P_2 being a film-based (CGH etc.) mask. We briefly note 3 optimization NNs and discuss their realization on Figure 1.

4.1 MIXTURE NN

In this case, the input signal $\underline{c}$ is the sum of a number of reference functions $\underline{k}^e$ (e = 1 ··· E references exist, plus unknowns)

$$\underline{c} = \sum_e x_e \underline{k}^e. \tag{1}$$

We desire to find the fractional amounts x_e of those references $\underline{k}^e$ present. This problem arises in the analysis of imaging spectrometry data [7] and in other cases. We minimize the MSE function $E_1 = \sum_n (c_n - \sum_e x_e k_n^e)^2$ and the constraint $E_2 = (\sum_e x_e - 1)^2$ that the sum of the x_e is unity. We also insure that all x_e satisfy $0 \leq x_e \leq 1$. The solution $\underline{x}$ using the neural evolution equation

$$\partial \underline{x}/\partial t \propto -\partial E(\underline{x})/\partial \underline{x} \tag{2}$$

can be written as the matrix-vector equation (for discrete time t) as

$$\underline{x}(t+1) = \phi[\underline{T}\,\underline{x}(t) + \underline{a}], \tag{3}$$

where ϕ satisfies $0 \leq x_e \leq 1$, the matrix $\underline{T} = \underline{K}^T\underline{K} + \underline{I}$ is fixed ($\underline{K}$ is the data matrix of the references, $\underline{K} = [\underline{k}^1 \cdots \underline{k}^E]$ and $\underline{K}^T\underline{K}$ is the VIP (vector inner product) matrix of reference data), and the vector $\underline{a} = \underline{K}^T\underline{c}$ is known (it varies with the input, but is fixed during the iterations in Eq.(3)). To implement (3) on Figure 1, we input $\underline{x}$ to P_1, and the fixed film mask P_2 is $\underline{T}$. We add $\underline{a}$ to P_3 (through Gate 1) and apply ϕ through Gate 2 to produce the new P_1 neuron values (linear input neurons are formed using Gate 4).

For this NN application, we have modeled the various error sources in the optical processor (with a random variable with a given standard deviation denoting various analog optical system accuracies and noise sources). We find [3] that the error in the P_2 weights is the most dominant error source (together with the uniformity of the P_1 illumination). With proper P_2 mask encoding and correction for P_1 illumination, sufficient accuracy exists. We have tested the algorithm with various mixtures of minerals (where k_n^e is the reflectance spectra of mineral e at various wavelengths n and $\underline{c}$ is the sum of several such reference signals). Table 1 shows the results obtained for only one element present (pure with both small and large grain size) and with mixtures of 10 different elements with different amounts of noise present in the composite input signal $\underline{c}$. The average number of iterations and the worst-case error in any x_e are noted.

4.2 MATRIX-INVERSION NN

This NN produces the inverse of a matrix directly [8]. This operation is vital for various real time phased array radar, signal processing, PR, and NN applications. Consider calculating the inverse of the matrix $\underline{Q}$ with elements Q_{ab}. The values of the neurons X_{ab} will denote the elements of $\underline{Q}^{-1}$. We solve this by minimizing the MSE function ϵ_1

$$\epsilon_1 = \sum_a \sum_c [\sum_b Q_{ab} X_{bc} - \delta_{ab}]^2. \quad (4)$$

When $\epsilon_1 = 0$, $\underline{Q}\,\underline{X} = \underline{I}$ and $\underline{X} = \underline{Q}^{-1}$. Substituting into the neuron state time evolution equation

$$\partial X_{ab}/\partial t = -\eta \partial E/\partial X_{ab} = -\eta[\sum_c \sum_d Q_{ca} Q_{cd} X_{db} - Q_{ba}] \quad (5)$$

and discretizing time δt with $\lambda = -n\delta t$ where n is the time index, we obtain

$$X_{ab}(n+1) = X_{ab}(n) + \lambda[\sum_c \sum_d Q_{ca} Q_{cd} X_{db} - S_{ab}]. \quad (6)$$

We write this as a matrix-vector equation

$$\underline{x}(n+1) = \underline{x}(n) + \lambda[\underline{T}\,\underline{x}(n) - \underline{s}] \quad (7)$$

where $\underline{x}$ is a vector of N^2 elements or neurons (the $N \times N$ elements of Q^{-1}), $\underline{T}$ is an $N^2 \times N^2$ interconnection matrix, and $\underline{s}$ is the 1-D list of the elements Q_{ab}.

From (7), we see that this NN formulation is similar to (3) for the mixture NN. It can thus be implemented on Figure 1 as before. The $\underline{x}(n)$ term on the right hand side of (7) can be included in the M-V product $\underline{T}\,\underline{x}$ by the addition of ones to the diagonal of $\underline{T}$. Thus the new $\underline{x}(n+1)$ values are given by a M-V product with the prior $\underline{x}(n)$ vector with a vector $\underline{s}$ subtracted from the result. The vector $\underline{s}$ subtraction is performed through Gate 1 with linear neuron values for the next n+1 time step produced using Gate 4. The only notable exception is that the matrix $\underline{T}$ weights now vary as a function of the matrix $\underline{W}$ being inverted (by comparison, the matrix in the mixture NN is fixed for a given database). The block Toeplitz structure of $\underline{T}$ allows for a very novel and efficient acousto-optic NN realization [8]. However, here we emphasize the realization of a variety of NNs on the same architecture (Figure 1) with a 2-D SLM (or film) at P_2.

Table 2 summarizes results obtained with various matrices. As seen, the number of iterations required is modest and the MSE accuracy of the resultant $\underline{Q}^{-1}$ inverse is generally within the 1% accuracy expected from an analog processor. This algorithm thus appears very attractive since round-off errors do not accumulate and a meaningful result (with 1% accuracy) is obtained with a 1% accurate analog optical processor.

4.3 MULTITARGET TRACKER (MTT) NEURAL NET

We devised, described, and simulated an MTT NN using only position sensor data. This system [9] resulted in a new NN that minimized a cubic energy function. The optical architecture required multiplication of a matrix (the vector outer product (VOP) of the present neuron state) times a tensor. Our algorithm and architecture reduced the required 2-D space bandwidth product for the tensor by a factor of over 1000. Although this cubic energy function NN algorithm is very attractive, it requires a new tensor for each new set of measurements. Instead of using measurement data over 3 time frames, we devised a new quadratic energy MTT NN using range and velocity sensor data. This is preferable, results in a simpler NN system using a fixed 2-D mask (rather than a real time high

bandwidth 2-D or 3-D SLM). Both of these NNs are measurement based, require fewer input neurons than other MTT NNs, do not require track initiation, nor a Kalman filter or extended Kalman filter post processor. We now describe our quadratic MTT NN algorithm, its realization on Figure 1, and initial data results.

We assume N_m measurements in two sequential time frames i and j. The objective of the processor is to assign each measurement at time i to a measurement at time j. We use N_m^2 neurons X_{ij}. We use the differences D_{ij} between all measurement pairs. The energy or error function to be minimized with respect to the neuron state X is

$$E(X) = C_1 \sum_i \sum_j X_{ij} D_{ij} + C_2 \sum_i (\sum_j X_{ij} - 1)^2 + C_3 \sum_j (\sum_i X_{ij} - 1)^2. \quad (8)$$

Term 1 is a minimum when measurements i and j in two time frames are the most similar (term 1 reduces the strengths of neurons associated with large D_{ij}). Terms 2 and 3 insure that for each measurement i (j) in frame 1 (2) there is only one measurement j (i) in frame 2 (1) associated with it. The weights C_1-C_3 are chosen to emphasize the term desired (dependent upon sensor properties, scenarios, etc.). We use the Hopfield neural evolution equation

$$X_{kl}(n+1) = X_{kl}(n) - \eta \Delta X_{kl} \quad (9)$$

where n is the discrete time index, η is the step size, and from (8)

$$\Delta X_{kl} = \partial E(X)/\partial X_{kl} = D_{kl} + 2(\sum_j X_{kj} - 1) + 2(\sum_i X_{ij} - 1). \quad (10)$$

We write (10) as the matrix-vector equation

$$\Delta \underline{X}_i = \sum_j \underline{\underline{M}}_{ij} \underline{X}_j + \underline{D}_i \quad (11)$$

and thus we can implement this algorithm on the system of Figure 1.

The vector $\underline{X}_j$ is the N_m^2 dimensional lexicographically ordered vectorized version of the X_{ij} neurons at P_1, $\underline{D}_i$ is the N_m^2 difference vector added to the P_3 outputs, and $\underline{\underline{M}}$ is a fixed (film-based) matrix at P_2 in Figure 1. As before, we add $\underline{D}_i$ to the P_3 output through Gate 1 (in Figure 1), we use Gate 2 to apply a nonlinearity to the output to insure $0 \leq X_j \leq 1$. The final P_1 neurons are now binary (Gate 4).

Figure 3 shows the typical $\underline{X}_{ij}$ neuron outputs (in 2-D, for ease of understanding) at different time steps in their evolution. The amount of area shaded in the 2-D outputs denote the neuron analog output. As seen, the final output has one "on" (value $\simeq 1$) neuron per row and column (i.e. one measurement pair assigned in each time frame). Excellent P_C = 100% performance has been obtained in noise for various scenarios with this algorithm [10].

5. SYMBOLIC AND PRODUCTION SYSTEM NNs

5.1 PRODUCTION SYSTEM NN

When a predicate calculus set of rules as "if-then" statements describes a production system, one can employ a NN to determine new true facts (activated output P_3 neurons) given true input facts (activated P_1 neurons) and rules (as P_1-P_3 interconnection weights). Figure 4 shows the NN for the simple set of 4 rules: A→B, a AND C AND F→G, B→C, F AND G→C. Each fact (A to G) is assigned to one input and output neuron. To implement this on Figure 1, true facts are denoted by activated P_1 neurons and new inferred facts are given by P_3 neurons that exceed threshold. The rules (weights) are a fixed P_2 matrix. The P_3 output is thresholded to produce binary neurons (Gates 2 and 4) for the next P_1 input. This produces new inferences and new rules not explicitly encoded.

5.2 SYMBOLIC CORRELATOR NN

We have interfaced a production system to a multichannel optical correlator to achieve a symbolic correlator [11]. This is a most unique NN, since it is the only NN that allows multiple objects to be handled in parallel with both shift and distortion invariance for scene analysis. Figure 5 shows the basic concept of this processor. The multi-channel optical correlator provides a multiple output (D-digit symbolic word) for each object present in the field of view (FOV). An optical correlator thus allows multiple objects to be handled and true shift invariance. The multichannel correlator used and its symbolic output allows many classes of objects to be identified (with D=4 channels or filters and L=10 output levels, over 10,000 object classes can be accommodated on one processor). The recent simulated [12] and real time [11] optical laboratory data we have obtained used filters to recognize the presence of various object parts, the encoding of these symbols was then fed to a production system NN. Excellent distortion-invariant and multiple object results were obtained [11].

6. ADAPTIVE LEARNING PATTERN RECOGNITION (PR) NN

Various multi-layer NNs can be produced. With three neuron layers, any piecewise nonlinear discriminant surface can be produced. Figure 6 shows a 3 layer NN with neuron layers P_1 (input), P_3 (hidden layer) and P_5 (output). The P_2 matrix in Figure 1 provides the P_1 to P_3 weights needed. We implement the P_3 to P_5 (second to third layer) neurons in Figure 6 in hardware through Gate 3 in Figure 1 (this is realistic with the new NN we consider).

One advantage of a NN over standard PR classifiers is its organized ability to produce nonlinear decision surfaces. We feel that a PR NN should utilize standard PR techniques where appropriate and NN techniques where they are preferable. A marriage of PR and NN techniques is preferable. Our NN (Figure 6) uses PR techniques (linear discriminant functions (LDFs) and clustering) to select the number of P_3 neurons and the initial P_1 to P_3 weights. NN techniques are then used to refine these initial weights. A hybrid PR-NN thus results. We consider a multiclass PR classification problem with one P_5 neuron per class (the activated P_5 neuron denotes the class of the input fed to P_1). The P_1 neuron representation space (Section 3) we use is a feature space with inherent shift and distortion invariance and with a low dimensionality. This provides shift and distortion-invariant multiclass PR. We select 2-5 neurons at P_3 per class using clustering techniques. These are example/prototype/or cluster neurons and hence we refer to this as an adaptive clustering neural net (ACNN). The initial P_1

to P_3 weights are selected using LDF methods. They are then refined using NN methods to produce nonlinear piecewise decision surfaces.

Most NN learning algorithms are such that they are not easily realized with optical processing. We envision the use of gating learning with learning being off-line (on the HNC digital hardware section of Figure 1) with the weights being fixed after learning. We consider classification of input data to be an on-line problem requiring real-time optical processing (on the P_1-P_3 optical system of Figure 1). Thus, we employ a hybrid optical-digital NN.

Our present purpose is to show how many different NNs can be realized on the architecture of Figure 1. Thus, we only briefly highlight our ACNN [13] realization on Figure 1 (or in general, a multi-layer NN). We consider only supervised learning. The input P_1 neurons are analog features, the P_1 to P_3 weights are chosen as LDFs and are subsequently modified (in training on the digital NN) to produce piecewise nonlinear discriminant surfaces. Our cluster selection method determines the number of P_3 neurons used and removes this variable from the NN. The P_1 to P_3 weights are analog and encoded on a fixed mask at P_2 of Figure 1 (after training). The P_2 mask can be adapted as gated learning proceeds. The most active P_3 neuron is selected (Gate 2 in Figure 1) and binary P_3 neurons result (with one P_3 neuron being the most active). The P_3 to P_5 weights are binary and perform a mapping of the P_3 cluster selected to the final class designation (the P_5 neuron activated). Only one pass through the system is required in classification. The P_1-P_3 neuron processing is optical. The P_3 nonlinearity (Gate 2 in Figure 1) requires a maximum selection operation. The P_3-P_5 processing is performed digitally (Gate 3 in Figure 1), since it is only a simple mapping and the number of P_5 neurons is small. Figure 7 shows one result from this ACNN for a 3 class problem with 2 features. The samples in each class are indicated by different symbols. The nonlinear decision surfaces produced by the ACNN are indicated. Such surfaces are necessary to separate these data samples. Over 98% correct recognition was achieved.

7. SUMMARY AND CONCLUSION

A general optical/digital NN architecture and its hardware were described. The multi-functional nature of the system was emphasized - with the same processor shown to be capable of solving a variety of NN problems. We have highlighted many of these uses. The system functions as an associative processor (AP). We specifically use it as a heteroassociative processor (HAP) for distortion invariant pattern recognition. We also employ it as a closure AP (operating on facts). The system is suitable for many optimization NNs. We have highlighted its use as a mixture processor, a multitarget tracker and a matrix inversion system. In the first 2 cases, an external vector is added to the M-V neuron output. Both analog and binary neurons are used (depending upon the application). Analog weights are used. The use of the system as a production system and a symbolic correlator NN were noted (this NN handles multiple objects in the field of view). Finally, its use in adaptive learning (distortion invariant PR classification) was discussed - where it functions as a general multi-layer NN capable of any piecewise nonlinear discriminant surfaces.

ACKNOWLEDGMENT

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency monitored by the U.S. Army Missile Command.

Input spectra	Noise percentage n (%)	Average number of iterations	Worst case error in any x_e (%)
Pure, large	0	738	3.39
Pure, small	0	1094	11.33
Pure, large and small	0	940	5.42
Pure, large	2.5	1155	5.16
Pure, large	5	1153	4.91
Mixture, large		2236	0.98
Mixture, large	2.5	3603	1.46
Mixture, large	5	3537	1.86

Table 1. Simulation results for the determination of the composition of an input element or mixture.

No. of Iterations	Accuracy of Result
55	1.28%
88	0.13%
111	0.01%

Table 2. Optical neural net matrix inversion solution data.

References

1. Proc. International Conf. on Neural Networks, San Diego, CA, July 1988.
2. Proc. International Joint Conf. on Neural Networks (IJCNN), Washington, D.C., June 1989.
3. D. Casasent and T. Slagle, "A Hybrid Optical/Digital Neural Network", Proc. SPIE, Vol. 1215, January 1990.
4. P. Vermeulen and D. Casasent, "Modulated Error Diffusion CGHs for Neural Nets", Proc. SPIE, Vol. 1211, January 1990.
5. B. Telfer and D. Casasent, "Ho-Kashyap Content-Addressable Associative Processor", Proc. SPIE, Vol. 1294, April 1990.
6. T. Kohonen, Self-Organization and Associative Memory, Springer-Verlag, New York, 1987.
7. E. Barnard and D. Casasent, "An Optical Neural Net for Classifying Image-Spectrometer Data", Applied Optics, Vol. 28, pp. 3129-3133, 1 August 1989.
8. E. Barnard and D. Casasent, "Optical Neural Net for Matrix Inversion", Applied Optics, Vol. 28, pp. 2499-2504, 1 July 1989.
9. E. Barnard and D. Casasent, "Multitarget Tracking with Cubic Energy Optical Neural Nets", Applied Optics, Vol. 28, pp. 791-798, 15 February 1989.
10. M. Yee, E. Barnard and D. Casasent, "Multitarget Tracking with an Optical Neural Net Using a Quadratic Energy Function", Proc. SPIE, Vol. 1192, November 1989.
11. D. Casasent, E. Botha, J.Y. Wang and R.C. Ye, "Optical Laboratory Realization of a Symbolic Production System", Proc. SPIE, Vol. 1295, April 1990.
12. D. Casasent and E. Botha, "A Symbolic Neural Net Production System: Obstacle Avoidance, Navigation, Shift-Invariance and Multiple Objects", Proc. SPIE, Vol. 1195, November 1989.
13. E. Barnard and D. Casasent, "Image Processing for Image Understanding with Neural Nets", *Proc. IEEE International Joint Conference on Neural Networks (IJCNN)*, June 1989, Washington, D.C., Vol. 1, pp. I-111-115.

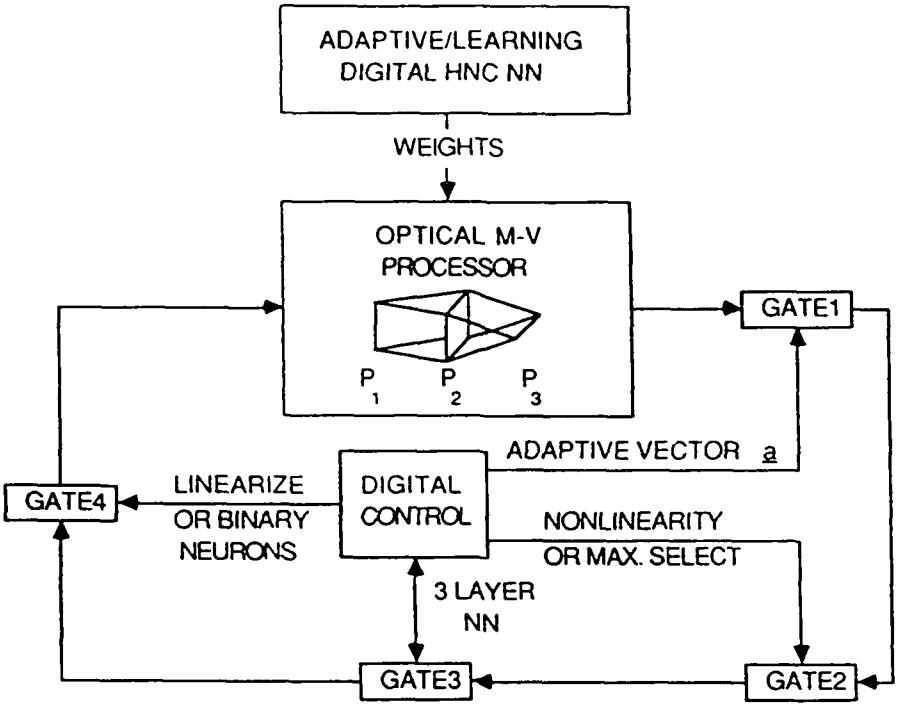


Figure 1. General hybrid optical/digital neural net.

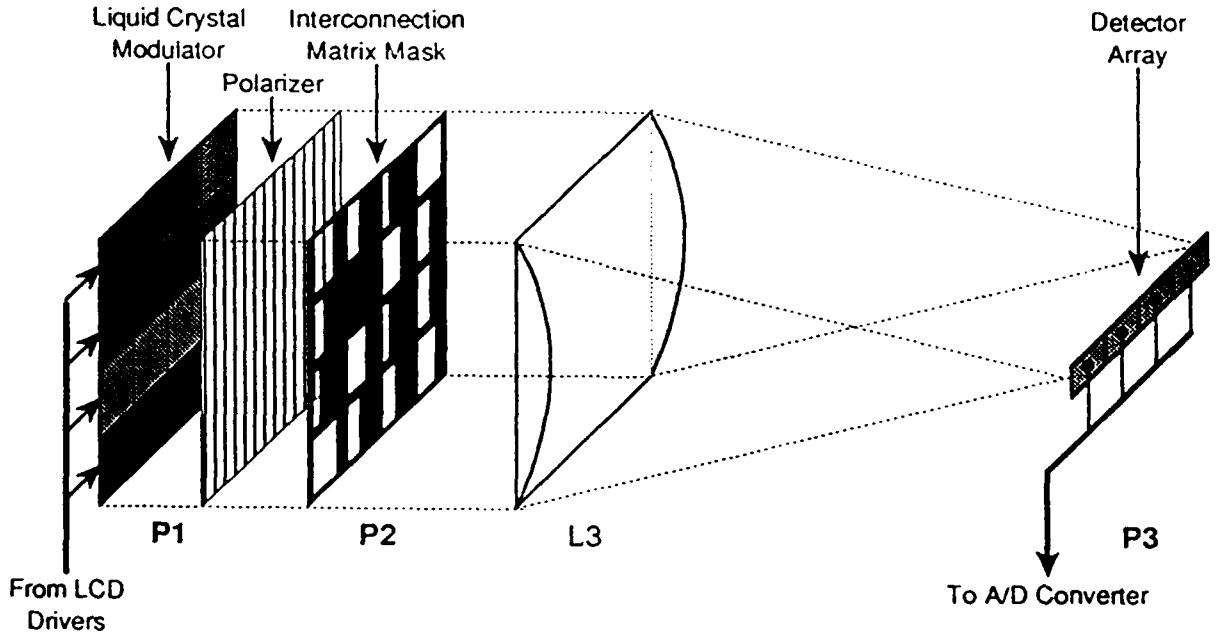


Figure 2. Simplified view of the optical matrix-vector multiplier.

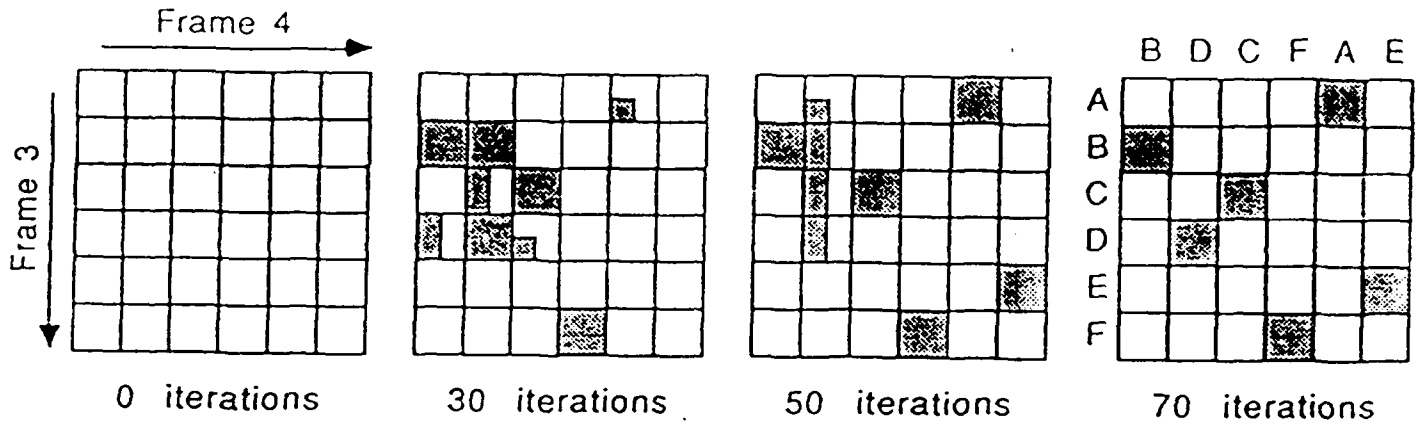


Figure 3. Typical MTT NN output neuron states at different states (iterations).

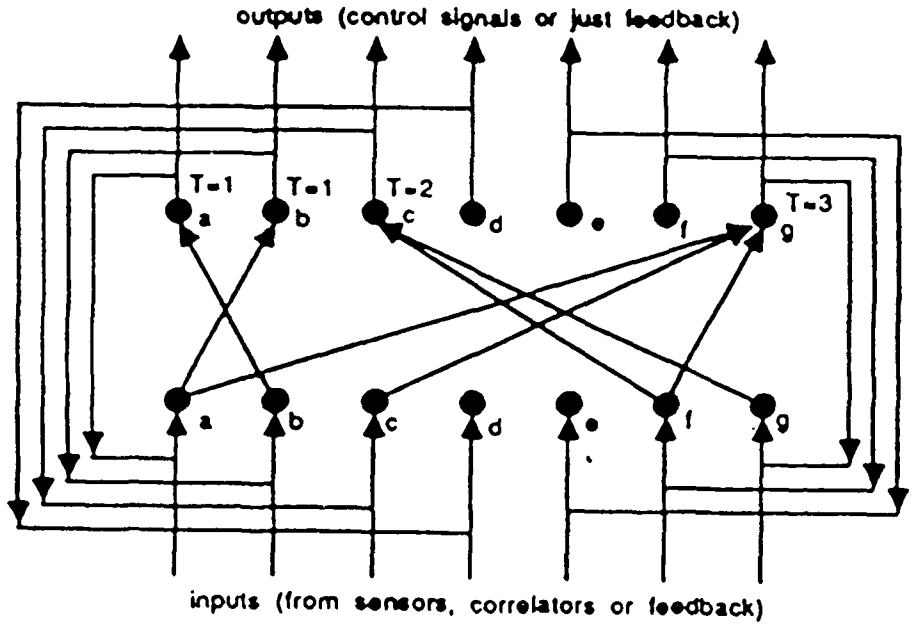


Figure 4. Representative production system NN.

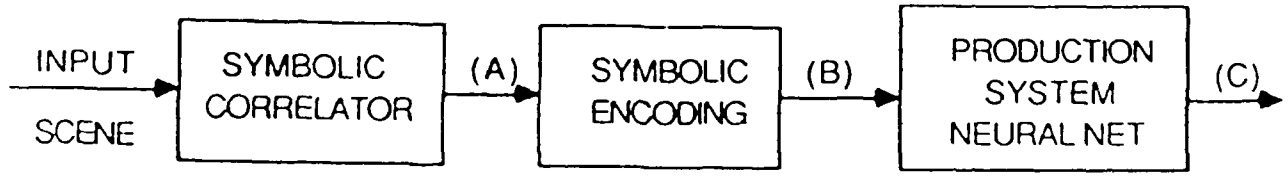


Figure 5. Symbolic correlator NN for handling multiple objects.

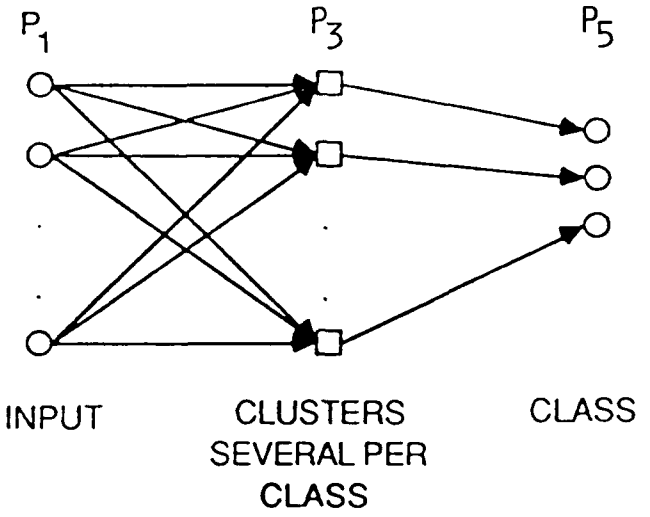


Figure 6. Three layer adaptive clustering NN (ACNN).

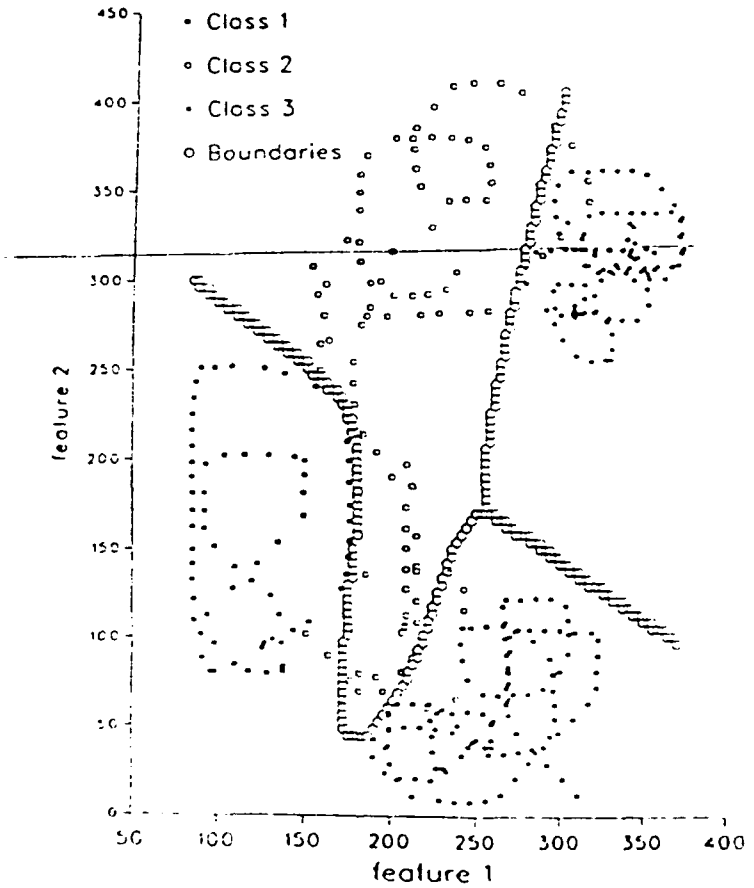


Figure 7. Representative piecewise nonlinear discriminant surfaces produced by our ACNN.

CHAPTER 3

"A Hybrid Optical/Digital Neural Network"

Proc. SPIE
Volume 1215
January 1990
Los Angeles

A HYBRID OPTICAL/DIGITAL NEURAL NETWORK

David Casasent and Timothy Slagle

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

A hybrid optical/digital neural net is described. Initial tests on the optical components are provided with the first simulated neural net results addressing various optical system error sources included. Attention is given to the accuracy required for each optical component, the dominant error source and the cumulative effect of multiple optical system error sources.

1. INTRODUCTION

Various neural net (NN) architectures and algorithms have been advanced. Several of these have been realized and tested in limited dimensionality and extent in the lab. In Section 2, we advance a new and most general purpose NN. It is a hybrid optical/digital NN (using a digital NN for training/learning and an optical N for on-line processing). Its wide usage in a multitude of applications will be detailed elsewhere [1]. Here we advance its basic concept and architecture (Section 2), we consider a specific optimization NN application (mixture analysis) in Section 3, and we provide the first simulation of optical NN error effects (Section 4). Prior NN simulations [2] have not been successful, due to an insufficient NN model and/or the choice of an NN architecture not easily lending itself to modeling.

2. HYBRID OPTICAL/DIGITAL NN ARCHITECTURE

Figure 1 shows the basic architecture we consider. It consists of a general-purpose hardware digital NN (the Hecht Nielson Corporation (HNC) Anza system) interfaced to an optical NN. The optical NN consists of an optical matrix-vector (M-V) multiplier. The vector data is fed to point modulators at P_1. The P_1 light is broadcast to uniformly illuminate different rows at P_2 (which contains the matrix data). The light leaving P_2 is integrated vertically onto a linear detector array at P_3. The P_3 output is thus the M-V product of the P_2 matrix data and the P_1 vector data. The optical M-V architecture is the basic element of the system. Its outputs (P_3) are processed in various manners (depending upon the application) before being fed back to the P_1 inputs. We also allow the P_3 output to be used to alter the P_2 matrix data (with an adaptive P_2 SLM used). The P_3 to P_1 digital feedback shown considers various NNs and the resultant processor is very general purpose [1].

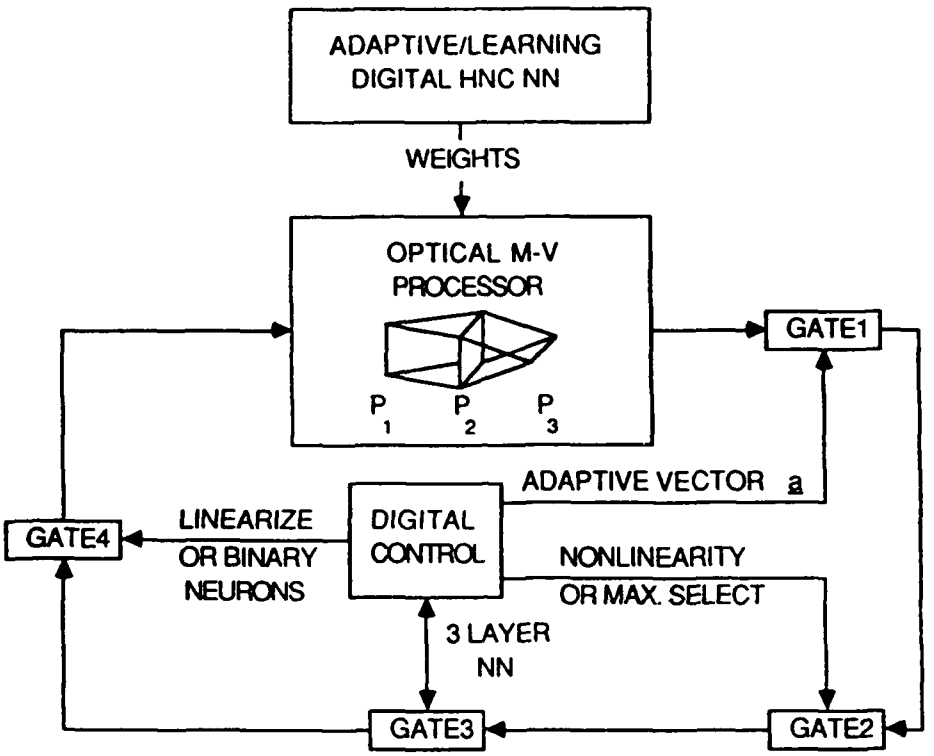


FIGURE 1. General hybrid optical/digital neural net.

3. CASE STUDY

The specific case study we consider is input 1-D data which is a mixture of several reference 1-D patterns. Specifically, for the case of input imaging spectrometer data, we consider the input to be the sum of the several reference spectra (for different minerals) present in a given input region. The objective is to determine which e of E spectra are present and the amount x_e of each that is present. A specific case study (such as this) is expected to quantify the spatial M-V system errors allowed and the individual component requirements.

The signal $\underline{c} = \{c_n\}$ received at each spatial region of the scene has N spectral components at the λ_n of the imaging spectrometer. This received signal is a mixture of the reflectance data $\underline{k}^e = [K_1^e \cdots K_N^e]$ for mineral element e for all λ_n,

$$\underline{c} = \sum_e x_e \underline{k}^e. \qquad (1)$$

The objective is to determine the elements e present ($e = 1 \cdots E$) and the fractional amount x_e of each. The $\underline{k}^e$ reflectances for E minerals are available and the data matrix $\underline{K} = [\underline{k}^1 \cdots \underline{k}^E]$ describes the reference data. We have spectra in $N = 128$ reduced bands for $E = 500$ elements. To solve (1) for $\underline{x}$, we consider a neural net (NN) solution. We write the MSE as one error term to be minimized

$$E_1(x) = 1/2 \sum_n (c_n - \sum_e x_e K_n^e)^2 \qquad (2)$$

where the K_n^e are the reflectance data at wavelength λ_n for element e. We also impose the condition that the sum of the x_e equal one

$$E_2(x) = 1/2(\sum_e x_e - 1)^2 \quad (3)$$

The solution $\underline{x}$ that minimizes the error or energy $E = E_1 + E_2$ is desired.

To describe this as an NN optimization problem, we denote the solution $\underline{x}$ by a neuron array and allow the neurons to evolve as

$$\partial \underline{x}/\partial t \propto -\partial E(\underline{x})/\partial \underline{x}. \quad (4)$$

With (4), E decreases monotonically with time and no local minima occur. Substituting E into (4), and using discrete time, we obtain

$$\underline{x}(t+1) = \phi[\underline{T}\,\underline{x}(t) + \underline{a}], \quad (5)$$

where $\underline{T} = -\eta \underline{K}^T \underline{K} + \underline{I}$, $\underline{a} = \eta \underline{K}^T \underline{c}$ and ϕ is a nonlinear function that satisfies the additional constraint

$$0 \leq x_e(t) \leq 1 \quad (6)$$

and $\eta = 2/\mathrm{Tr}[\underline{K}^T \underline{K}]$ as in the Widrow Huff LMS algorithm.

We will compare the evolution solution in (5) to the pseudoinverse solution

$$\underline{x} = (\underline{K}^T \underline{K})^{-1} \underline{K}^T \underline{c} = \underline{K}^+ \underline{c} \quad (7)$$

which does not satisfy (6) to show that a NN solution is needed. The neural net solution in (5) can be achieved on the optical system of Figure 1 as we now discuss. The matrix $\underline{T}$ is placed at P_2 (it is fixed and film can be used for it - in this application - and in most optimization NNs). The P_1 outputs are $\underline{x}$ and at P_3 we obtain $\underline{T}\,\underline{x}$. Since $\underline{K}$ is fixed, we form $\underline{a}$ in digital hardware. Thus, in Figure 1, the M-V multiplication is performed optically and the external vector is calculated digitally and added to the optical M-V result. This output is then thresholded and fed back to P_1. The feedback in Figure 1 achieves this.

3. OPTIAL NN FABRICATION

For the present optimization NN, the matrices are fixed. This is the case for nearly all optimization NNs. Thus, we consider the use of film for the matrix P_2 data. We detail elsewhere [3] how to optimally encode this matrix data on film. Figure 2 shows the optical M-V processor in more detail and Figure 3 shows its electronic support. In Figure 3, the P_1 neurons are formed from a 2-D liquid crystal (LC) display [4] modified with all elements in a row fed with the same signal. The display has 20 rows of 40 elements (4.2×8.4 cm^2), each LC pixel is 2.1×2.1 mm^2. We employ it as a set of 20

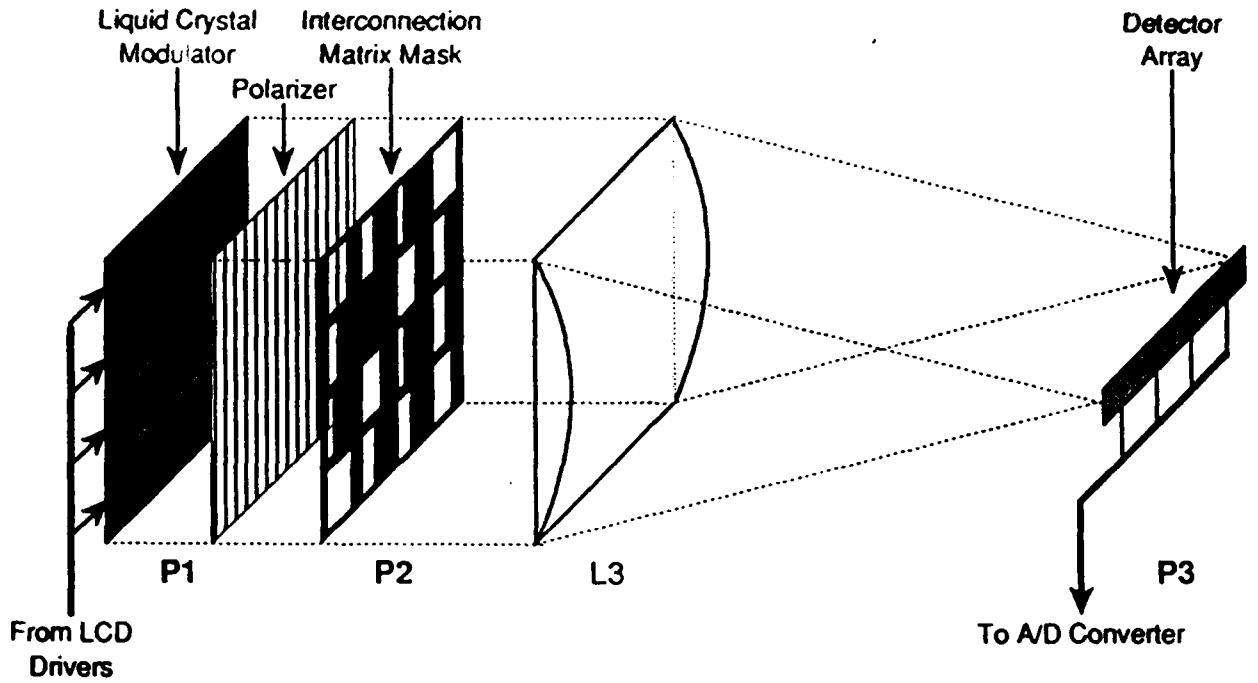


FIGURE 2. Simplified view of the optical matrix-vector multiplier.

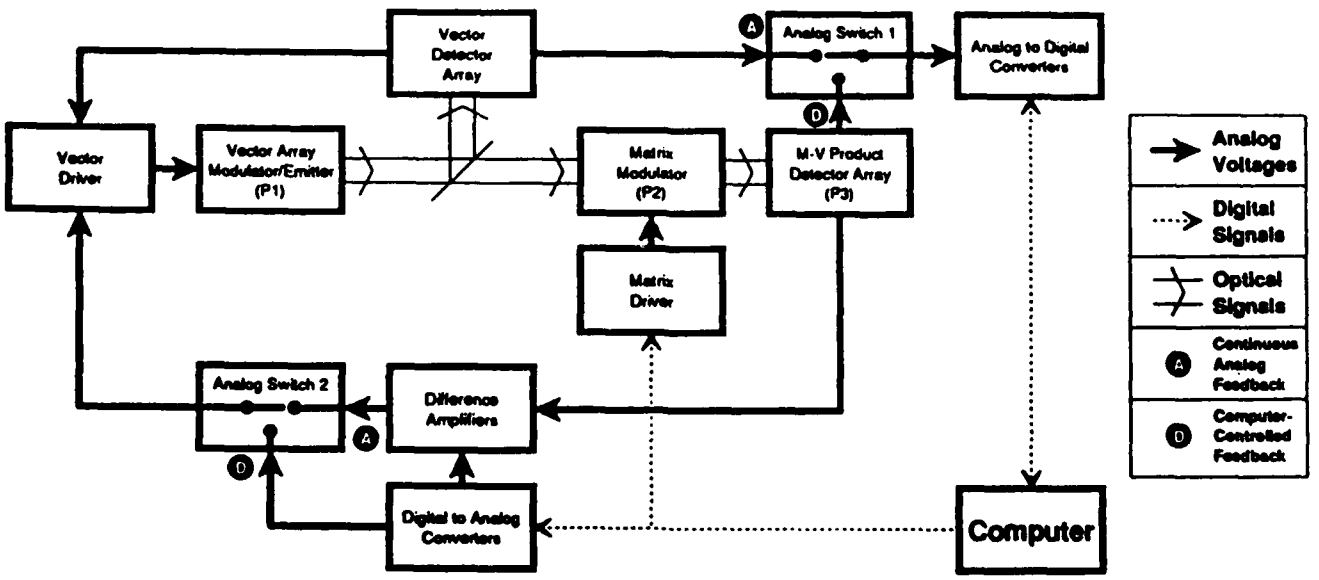


FIGURE 3. Block diagram of laboratory system.

stripe modulators. The device has TN LC material sandwiched between two glass plates, whose surfaces adjacent to the LC material are coated with etched transparent conductors to form electrically isolated elements. At present, we use a set of 4×4 LC pixels (8.2×8.2 mm^2) in the center of the display. This stripe P1 design is attractive since it avoids the need for external collimating and imaging optics between P1 and P2. It allows P2 to be placed in contact with P1, thus simplifying design. P2 data is recorded as a binary computer generated hologram (CGH) pattern using a new encoding technique that provides high accuracy [3].

A HeNe laser (λ = 633 nm) serves as the present light source illuminating the LC at P1. The initial optics illuminating P1 have been designed. We insure uniform illumination of P1. The beam radius R required to produce a uniformity X = 0.995 (0.5%) over a diameter D = $\sqrt{2}$ (8.4) = 11.88 mm is R $\simeq$ $D/2(1-X)^{1/2}$ = 84 mm. This will pass only 0.5% of the light. This is sufficient with a 30 mW laser, but can be improved by use of laser diode sources and CGHs. The initial P2 to P3 optics have been designed using two cylindrical lenses (L2 with f_{L2} = 25 cm and L3 with f_{L3} = 15 cm).

At P3, we use TRW OP509SLC phototransistors with a plastic lens case. The detectors have a 1.3 mm diameter active area and are on 2.54 mm centers. The light from each 2.1 mm wide LC and P2 column is magnified by 2.54/2.1 = 1.21 to image onto the detectors. With 1.3 mm detectors, the width of a P2 element can be no larger than 1.3/1.21 = 1.07 mm. We use 0.9 mm wide P2 elements to allow an 0.17 mm guard band horizontally. We use 1.8 mm of the 2.1 mm height of each element (an 0.3 mm guardband).

Thus, each P2 element has an active area of 1.8×0.9 mm^2 (in the 2.1×2.1 mm^2 area). The Linotronics recorder we use to produce the P2 mask has 20 μm diameter spots on 10 μm centers. Thus, in the 1.8×0.9 = 1.62 mm^2 area of one element, we can record 4050 dots or weights with 4050 gray levels using dot corrected CGH techniques [3].

We have fabricated the major portion of the support electronics (Figure 3). The vector driver (one op amp and CMOS SPDT switch per P1 input element) provides the ac zero-mean square wave required by the LC. The input is a dc voltage and the output is a zero-mean squarewave with a peak-to-peak amplitude that is twice the unipolar input. The circuit can operate at 1 KHz (this is much faster than the 20 Hz frame rate of the present LC). We have also fabricated the detector amps (transimpedance op amps LF412). The gain of each is individually adjusted to correct for variations in channels and detector efficiencies. The max output is 5.0 volts at the maximum expected light level (a 220 kΩ feedback resistor is used).

The data acquisition and generation system is now described. In this section, the outputs from the P3 detector amps are A/D converted, fed to an IBM PC/AT whose outputs are D/A converted, fed to a demultiplexor S/H (sample and hold) circuit before being fed in parallel to the P1 vector drivers. All analog signals are 0-5 V. All circuits use +5 and $\pm$7.5 V power supplies. This system consists of an IBM PC/AT, a PC-Bus data acquisition board, and special demux S/H circuitry. The PC/AT runs at 6 MHz, has 512 kB of memory and a 30 MB hard disk. Control software is written in Turbo C and 80286 assembly code. It computes the input drive to P1 and outputs these signals. It also digitizes the output P3 neuron data. The data acquisition board is a Data Translation DT2821. It contains one 12-bit 50 kHz A/D and two independent 12-bit 130 kHz D/As. A 16-channel multiplexor inputs the parallel P3 detector data to the A/D. The board also contains 16-bits of TTL level signals for later I/O

control use. Presently, we digitize 4 detector outputs in 80 μs (50 kHz). The calculated P1 inputs for the next cycle are sequentially D/A converted, fed to a demultiplexor S/H (an LF398 S/H circuit) to allow them to be fed in parallel to the P1 drivers. Four parallel S/H circuits are presently used. The D/A can cycle through a 32-element vector in 2.5 ms (7.7 μs per input or 30.8 μs for our present 4 inputs).

The noise level of the P3 detectors were measured to be 10 mV out of 5 V for a $\sigma_n = 2\times10^{-3}$. We measured other parameters of our system for use in simulations of error source effects. The contrast ratio of the LC at P1 was measured to be 5690:1. This is much larger than the value typically reported, due to the type of drive signal we use in our use of the device. We drive each modulator element with a continuous signal. Conventionally, one applies a pulse voltage to one element and then returns (time-multiplexed) to it later. The speed of the LC device was measured by applying a 2 Hz square wave signal with various amplitudes. The results (Figure 4) show that a 50 ms switching time (average of rise and fall times) results for drive voltages above 7 V RMS. Increasing the drive voltage decreases LC rise time and increases fall time. All times are measured from the 10-90% points. We linearize the LC transfer function by the 512 point table look-up. The results (Figure 5) show a 512 point linearity is obtained.

4. OPTICAL NN SIMULATION

We now provide the first models for the accuracy required in the various elements of any optical NN. We consider our NN for our mixture NN application. We also compare the pseudoinverse solution and show that a NN solution is required. We consider 4 mixture case studies with different numbers of elements considered and with different numbers of non-zero elements. Case 1 (4 elements, 2 non-zero), Case 2 (4 elements, all non-zero), Case 3 (8 elements, 4 non-zero), and Case 4 (8 elements, all non-zero). Various amounts of elements were added, with the amount of each varied. The elements used were taken from the 20 most common minerals. Noise (zero-mean, Gaussian) was added to the data. The average of 10 runs (Cases 3 and 4) or 20 runs (Cases 1 and 2) were used for each x_e set (with different noise realizations). Each data point also represents the average of 10 x_e choices. Thus, each data point is the average of 100 or 200 different runs.

4.1 Pseudoinverse versus Neural Net Solutions

When x_e has most values near 0 or 1 (Cases 1 and 4) as occurs in all practical cases, we expect the NN solution to be better. Figures 6 and 7 show the data obtained for the 4 cases with different amounts of input noise (SNR). We see that the average neuron error is much less for all NN cases (dashed lines) than for the pseudoinverse (solid line) cases and that the difference is more at lower input SNR and for cases with more zero-valued elements (Cases 1 and 3). The number of iterations required also differs. With SNR = 20 dB, Case 3 required 50,000 iterations versus 1000 for Case 1 (due to the larger condition number). Mixtures with zero-valued x_e require more (1000 versus 130) iterations to converge. We started the NN iterations from the x_e calculated from the pseudoinverse solution (this is a useful new technique). Without this starting x_e value and using an arbitrary initial $\underline{x}$, we needed 8000 versus 1000 iterations.

We now discuss the significance of an average neuron error of 0.02 (our acceptable level). For Case 1, with 2 non-zero x_e, the average x_e = 0.50 and an average error of 0.02 is an 0.02/0.50 = 4%

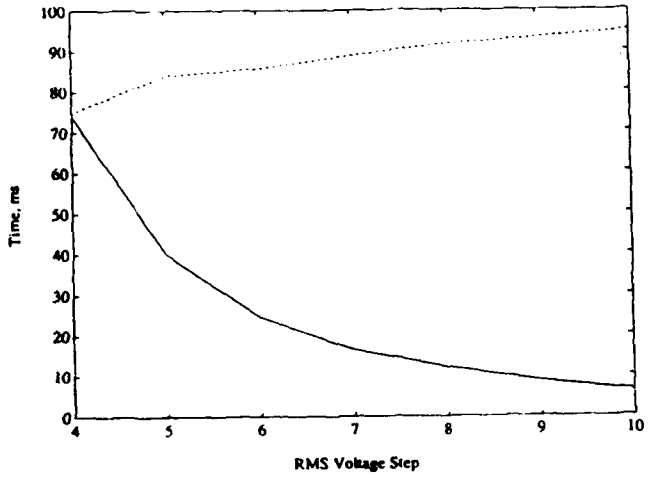


Figure 4. Rise (solid), fall (dashed), and switching time (dotted) vs. voltage.

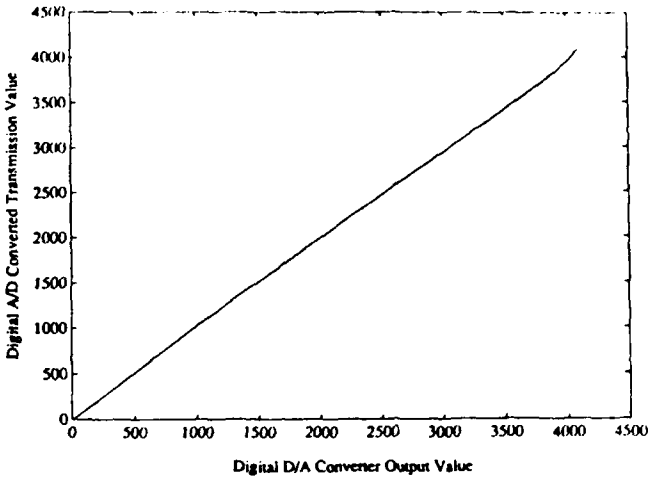


Figure 5. Transmission vs. digital output with look-up table.

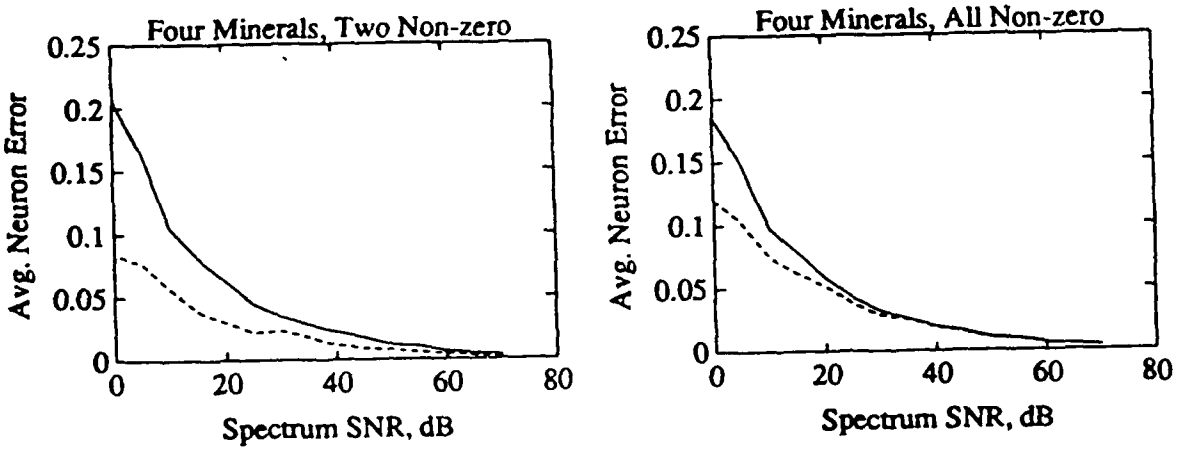


Figure 6. Accuracy vs. noise for pseudoinverse (solid line) and neural net (dotted line).

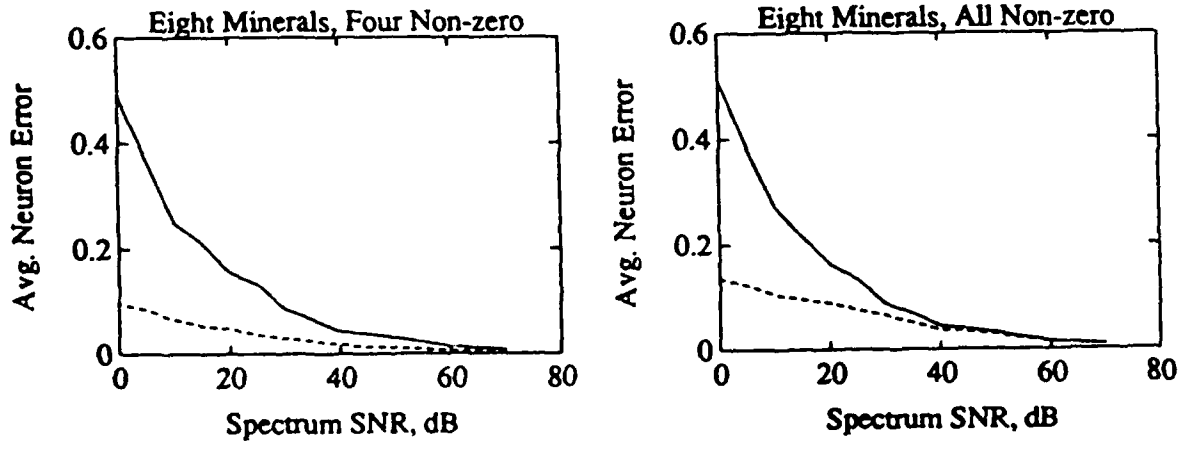


Figure 7. Accuracy vs. noise for pseudoinverse (solid line) and neural net (dotted line).

error. For Case 3, the average x_e = 0.25 and 0.02 is an average error of 0.02/0.25 = 8%. In practice, we expect few non-zero elements and thus Cases 1 and 3 are the most representative ones. They have a lower (better) average error (Figures 6 and 7). The choice of 0.02 is also a good goal since imaging spectrometer calibration accuracy is 5-7%.

4.2 Simulation Model

We model the NN in Eq. (5) with errors as

$$\underline{x}(t+1) = \phi[\underline{x}(t) + \eta[\underline{a} + \underline{n}_0 + \underline{n}_0 + \underline{n}_x(t) + \underline{N}_g(\underline{N}_m + \underline{K}^T\underline{K})\underline{x}(t)]. \quad (8)$$

We now discuss the errors included in (8). The detector P3 noise is additive, zero-mean, Gaussian and has a standard deviation $n_x(t)$ that is time-varying and uncorrelated (a new noise realization is used for each iteration). The additive P3 offset in the detectors (dark current) and detector differential amps plus A/D quantization noise are modeled by $\underline{n}_0$. The P3 neuron gain variations are modeled by $\underline{N}_g$. They are multiplicative and signal dependent. We assume negligible P1 errors (Figure 5 confirms this, since P3 feeds P1). To include N_g errors at P1, we would multiply the entire right hand side of (8) by another $\underline{N}_g$ factor. The P3 neuron errors $\underline{N}_g$ equivalently also include the effect of $\underline{N}_g$ errors in P1 neurons, thus we do not add the extra $\underline{N}_g$ factor. We represent neuron gain errors by a diagonal matrix $\underline{N}_g$ with diagonal elements $1+\underline{n}_g$ (where $\underline{n}_g$ are the gain variations). This handles a multiplicative error (1 + error) times a vector. Errors in the connection matrix at P2 are represented by the additive matrix $\underline{N}_m$ of random uniformly distributed values added to $\underline{T}$. This also includes errors in the uniformity of the light incident on P1. P3 offsets can be reduced by adjusting the P3 output amplifiers. Detector and P1 gain variations can also be adjusted by varying individual amp and drive circuits. P1 offset and nonuniform input light effects can be corrected within the P2 mask. Thus, all errors we consider are residual. Our goal is to determine the dominant errors, the level to which each must be reduced, how multiple errors combine and the performance expected for a given set of components with given specifications.

In all cases, we used a fixed convergence threshold of 10^{-4} (i.e. the largest element in $\underline{a}-\underline{T}\,\underline{x}$ must be $\leq 10^{-4}$) to stop iterations. When the noise added is above 10^{-4}, we average the last 20 correction vectors and stop iterating when the average is less than or equal to $3.25\underline{n}/\sqrt{20}$. This level increases with the standard deviation $\underline{n}$ of the noise source.

4.3 Error Source Results

Figures 8-11 show the effects of the four error sources separately. From Figure 9, we see that the effect of the additive detector noise is negligible, i.e. the average error is less than the standard deviation of the noise. The expected detector noise measured was $\sigma_x = 2\times10^{-3}$ and as seen, the average neuron error at this σ_x is much less than 10^{-3} and hence is negligible. Figure 9 shows the effect of offset error $\underline{n}_0$. To achieve an average error of 0.02, we require an offset variation of $\pm$0.0025 (for the 4 neuron cases) and $\pm$0.0013 (for the 8 neuron Case 3). With the 12-bit D/A and A/D, we expect a quantization error of $\pm1.2\times10^{-4}$. The detector op amps have 2 mV offset out of 5 V

or $\pm 4\times10^{-4}$. The expected total $\underline{n}_0$ is thus $(2.4+8)\times10^{-4} \simeq 10^{-3}$. At these $\underline{n}_0$ levels (10^{-3}), the average error is negligible (1.4×10^{-4} for case 4).

The effect of $\underline{N}_g$ errors is more significant (Figure 10). To keep the error below 0.02, we require the gain to be uniform within 1.25×10^{-3} (Case 2) or 5.5×10^{-4} (Case 3). For our lab system, the detector gains can be matched to 10^{-3} and the detector amps are matched to 10^{-4}. The additive N_m matrix errors are the most dominant errors (Figure 11). This is expected since it alters the problem and the energy surface and the required matrix accuracy increases with the condition number of the matrix $\underline{T}$. We require an accuracy of $\pm1.8\times10^{-4}$ ($\underline{N}_m = 3.6\times10^{-4}$) for Case 1 and $\pm10^{-4}$ ($\underline{N}_m = 2\times10^{-4}$) for Case 3. We can record matrix elements with an accuracy $\underline{N}_m = 1/4050 = 2.5\times10^{-4}$ and thus expect acceptable results. A beam uniformity of 0.005 incident on P1 will yield unacceptable 0.1 average errors. To achieve this amount of uniformity, we must correct with the P2 mask to achieve acceptable results in all cases.

We combined all noise sources and found that they add in an RMS fashion and that the mask accuracy and P1 beam uniformity are the critical parameters.

Acknowledgment

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency monitored by the U.S. Army Missile Command.

References

1. D. Casasent, "A Multifunctional Hybrid Optical/Digital Neural Net", Proc. SPIE, April 1990, to be submitted.
2. R.C. Frye, E.A. Rietman, C.C. Wong and B.L. Chin, "An Investigation of Adaptive Learning Implemented in an Optically Controlled Neural Network", *Proc. IEEE International Joint Conference on Neural Networks (IJCNN)*, June 1989, Washington, D.C., Vol. II, pp. II-457 - II-463.
3. D. Casasent and P. Vermeulen, "Modulated Error Diffusion CGHs for Neural Nets", Proc. SPIE, Vol. 1211, January 1990, to be submitted.
4. Ohio Arts Electronic Etch-a-Sketch Toy.

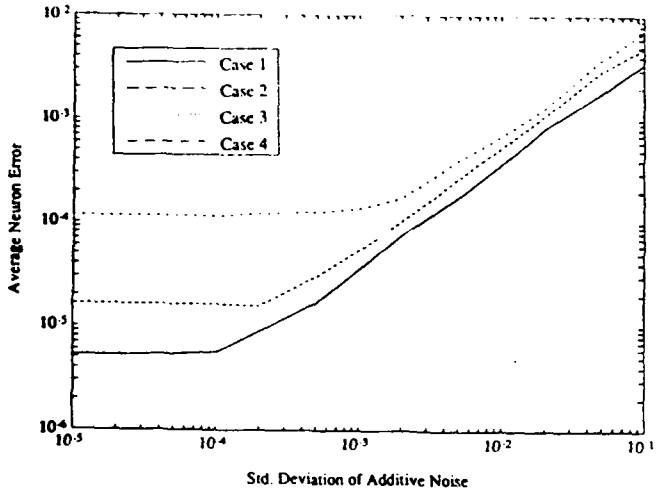


Figure 8. Average error vs. noise $\underline{n}_x$ in neuron outputs from P3.

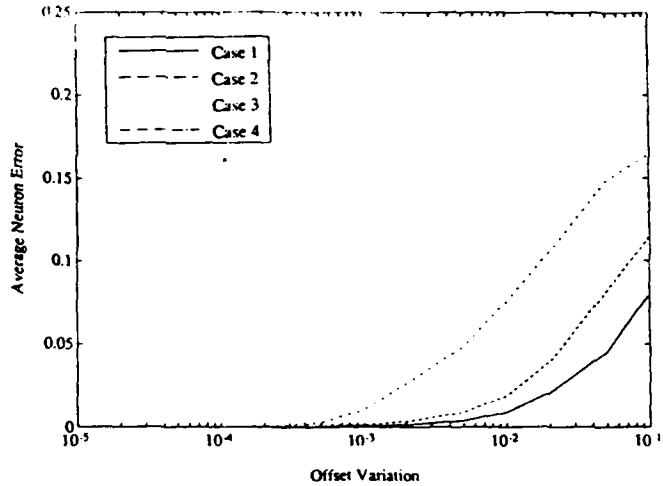


Figure 9. Average error vs. offset $\underline{n}_o$ in neuron outputs from P3.

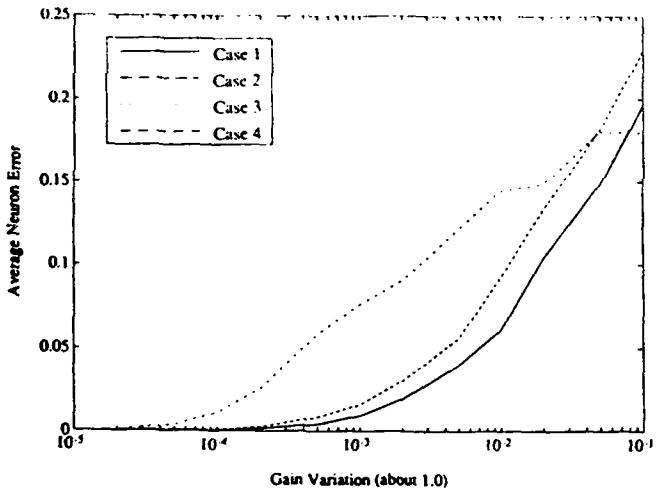


Figure 10. Average error vs. neuron gain variation N_g.

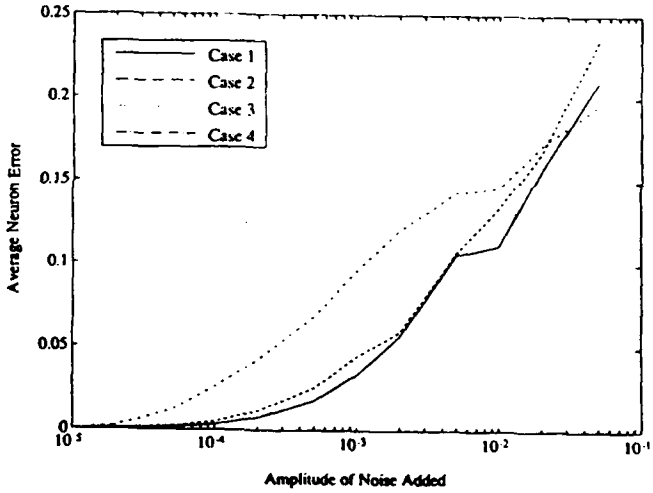


Figure 11. Average error vs. noise N_m in matrix values.

CHAPTER 4

"An Optical Neural Net for Classifying Image-Spectrometer Data"

Optical neural net for classifying imaging spectrometer data

Etienne Barnard and David P. Casasent

A problem in surface mineralogy is addressed; namely, how does one determine the composition of a mixture from its spectrum? A neural net algorithm arises naturally, and we detail the state equations of this net. An optical architecture and simulation results are presented.

I. Introduction

We consider the following problem: Given the spectra of a number of elements, determine the composition of an unknown input mixture from its measured spectrum. This problem has been studied in the context of surface mineralogy,[1] and conventional digital algorithms for solving it have been proposed.[2] These algorithms are fairly slow, since serial computation is used. Neural nets[3,4] are ideally suited for applications such as this one, because of the high parallelism achievable with them (as we will show). Thus, we sought to express the determination of the compositions as a problem suitable for solution by neural nets; this was done using the Hopfield minimization procedure. It is preferable that one implement neural nets with hardware capable of achieving their high degree of connectivity. Therefore, many researchers have investigated optical implementations of these nets.[5] We also consider an optical architecture to implement our algorithm.

In Sec. II, a mathematical description of the problem is developed. This is utilized as a basis for a neural network solution and an optical implementation, described in Sec. III. In Sec. III we also consider an alternative approach to our neural algorithm. Initial simulation results are presented in Sec. IV, and Sec. V summarizes our results.

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 7 June 1988.
0003-6935/89/153129-05$02.00/0.

II. Mathematical Description

Let $K^e(\lambda)$ denote the spectrum of mineral e, where λ denotes wavelength. There are E such minerals, so that e ranges from 1 to E. We shall discretize the spectra, so that they are measured at the N wavelengths λ_n, $n = 1,\ldots N$. Then each possible mineral is described by a vector $\mathbf{k}^e = (K_1^e, K_2^e, \ldots, K_N^e)$, where $K_n^e = K^e(\lambda_n)$. Similarly, the spectrum of the input mixture is described by a vector $\mathbf{c} = (c_1, c_2, \ldots, c_N)$. Our objective is to decompose the unknown input mixture into known elements (i.e., to determine the fractional amount x_e of each basic mineral present). This is given by the vector $\mathbf{x} = (x_1, x_2, \ldots, x_E)$, where $0 \le x_e \le 1$ and $\Sigma_{e=1}^{E} x_e = 1$. For the mixture described by $\mathbf{x}$, the spectral response at wavelength n is $\Sigma_e x_e K_n^e$. The difference between this vector and the measured spectral vector is a measure of how well $\mathbf{x}$ describes the input mixture. One can form a variety of scalar measures from this. The simplest such scalar measure is the Euclidean distance. As we shall see below, this is also the correct measure to use from probabilistic considerations. We thus determine how well a mixture vector $\mathbf{x}$ describes an input spectrum by considering the error measure

$$\epsilon = \sum_n \left(c_n - \sum_e K_n^e x_e \right)^2 . \tag{1}$$

This error is zero if the composition vector $\mathbf{x}$ describes the measured spectrum exactly. Otherwise, it is greater than zero. To determine the composition of an unknown input mixture, we must minimize this error with respect to $\mathbf{x}$. This ensures that the best possible match between the measured and predicted spectra is obtained. This minimization procedure can be viewed as a maximum-likelihood determination of the composition of the mixture, if we assume that the difference between the measured spectrum and the actual spectrum (as determined by the composition) is due to normally distributed zero-mean noise. Then

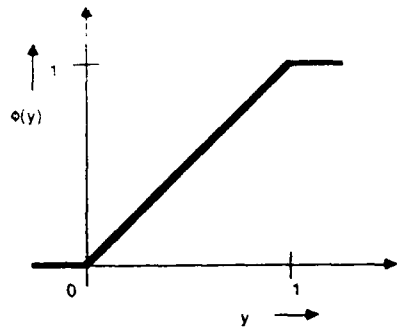


Fig. 1. The function ϕ.

the maximum-likelihood estimation of the composition of the mixture is given[6] by the minimum of Eq. (1).

III. Neural Network Description

This problem is not amenable to a conventional pattern-recognition solution, because there are no predetermined classes into which we can classify the input spectral data. That is, any combination of minerals has to be considered. On the other hand, the mathematical description derived in Sec. II is suitable for a neural network implementation. The application of neural nets to minimization problems is well established.[7]

The general procedure is to describe the minimization problem as the minimization of an energy function E, which depends on a set of variables x_j, where $j = 1,\ldots J$. There should be no difficulty in distinguishing between the energy and the number of elements; thus we retain the same symbol E for both. The objective is to find the set $\{x_j\}$ that minimizes E. A set of neurons is employed, with one neuron representing each of the variables x_j. To minimize E, we introduce a discrete time variable t, and design a neural network that evolves the neuron activities in time according to

$$x_j(t+1) = x_j(t) - \eta \frac{\partial E}{\partial x_j(t)}, \tag{2}$$

where η is a parameter that controls the speed of convergence. It is easy to show that E in Eq. (2) decreases as time progresses,[7] and that the net reaches a stable steady state when E attains its minimum value, since only then is no further decrease possible.

In our imaging spectrometer application, the minimization variables are the composition fractions x_e, and the energy function is a modified version of the error defined in Eq. (1). This modification is necessary because nothing in Eq. (1) forces the sum of the fractions to one. This constraint is enforced separately in the usual way be adding a positive semidefinite term to Eq. (1). This term attains its minimum value of zero when the fractions x_e add to one. The simplest term to add is $A(\Sigma_e\, x_e - 1)^2$, where A is a positive constant. This is minimized when the sum of the fractions x_e equals one. Thus, the energy function for our application is

$$E = \left(\frac{1}{2}\right)\left\{\sum_n \left(c_n - \sum_e K_n^e x_e\right)^2 + A\left(\sum_e x_e - 1\right)^2\right\}. \tag{3}$$

The factor of 1/2 is included for later convenience (the problem remains unchanged if the whole energy function is scaled by a constant factor). The constant A weighs the relative importance of the two energy terms in Eq. (3); it must be chosen large enough that $\Sigma_e\, x_e \approx 1$ for all states with low energy. Inserting Eq. (3) into Eq. (2), the evolution equation for our neurons becomes

$$x_f(t+1) = x_f(t) + \eta\left\{\sum_n \left[c_n - \sum_e K_n^e x_e(t)\right] K_n^f - A\left[\sum_e x_e(t) - 1\right]\right\}, \tag{4}$$

where the indices e and f refer to different neurons.

To rewrite this equation in matrix-vector notation, we note that the set $\{\Sigma_n\, K_n^e K_n^f\}$ forms an $E \times E$ matrix with horizontal index e and vertical index f. Its entries are the vector inner products of the spectra for minerals e and f. Similarly, the matrix-vector product $\Sigma_n\, K_n^f c_n$ is a vector with E entries. Using matrix-vector notation, Eq. (4) can be written as

$$\mathbf{x}(t+1) = \mathbf{x}(t) - \mathbf{T}\mathbf{x}(t) + \mathbf{a}, \tag{5}$$

where the neuron states are now a vector $\mathbf{x}$. The interconnection matrix $\mathbf{T}$ has elements

$$T_{ef} = \eta\left(\sum_n K_n^e K_n^f + A\right) \tag{6}$$

and is formed by adding A to every element of the matrix $\{\Sigma_n\, K_n^e K_n^f\}$ and multiplying by η. The summation over e in Eq. (4) is achieved by the matrix-vector product in Eq. (5). The vector a has components

$$a_f = \eta\left(\sum_n K_n^f c_n + A\right). \tag{7}$$

To obtain this from the vector $\Sigma_n\, K_n^f c_n$, we add A to every component and multiply the resulting vector by η.

One point has been neglected so far: nowhere in our neural description have the fractions x_e been forced to lie in the range [0,1]. Even though we minimize E when $\Sigma_e\, x_e = 1$, we need to ensure that each neuron activity x_e is positive and less than one, since it represents a fraction. This constraint is enforced by applying a nonlinear operator ϕ to Eq. (4), where

$$\phi[y] = \begin{cases} y \text{ for } y \in [0,1] \\ 0 \text{ for } y < 0 \\ 1 \text{ for } y > 1. \end{cases} \tag{8}$$

Figure 1 shows the value of this nonlinear operator as a function of its input y. If a neuron change would increase x_e beyond one, we set $x_e = 1$; similarly, if x_e would decrease to a value less than zero, x_e is set equal to zero. Thus the neuron evolution equation becomes

$$x_f(t+1) = \phi\left[x_f(t) - \sum_e T_{fe} x_e(t) + a_f\right]. \tag{9}$$

Thus, the steps involved in updating $\mathbf{x}(t)$ [to obtain $\mathbf{x}(t+1)$] are:

1. calculate the matrix-vector product $\mathbf{T}\mathbf{x}(t)$,
2. subtract this from $\mathbf{a}$,

3. add the resulting vector to the previous neuron state vector **x**, and

4. threshold as prescribed by the function ϕ.

An optical implementation of Eq. (9) is shown in Fig. 2, where P_1–P_3 is a standard optical matrix-vector multiplier, which multiplies the matrix M_1 at P_2 by the vector at P_1. The matrix M_1 is the matrix T. It is fixed by the spectral properties of the E known minerals, and can be recorded on a film mask (M_1 in Fig. 2). Since A and the spectral measurements K_n^e are greater than or equal to zero, all the entries in T are positive. The intensity of LED e in array LED_1 is proportional to $x_e(t)$. The light from the LEDs in P_1 is expanded horizontally by lens L_1, so that each LED illuminates a row of M_1. Lens L_2 integrates (vertically) the light leaving the columns of M_1.

The vector a depends on the measured spectrum of the input mixture to be analyzed, but does not change from one iteration to the next. It consists of two parts as in Eq. (7). The first term, a matrix-vector product, is calculated on the P_4 to P_6 optical system. The second term, a constant bias, is added after the detector array (Det) at P_6. The matrix at P_5 is also fixed (it is $E \times N$, and consists of the spectra of the E minerals). Thus M_2 can be a film mask, while the input to the LED array LED_2 is proportional to the spectral samples of the measured mixture to be analyzed. The outputs from the two linear detector arrays at P_3 and P_6 are subtracted in electronics to form $\mathbf{a} - \mathbf{Tx}$. Alternatively, the vector a can be calculated electronically, since it does not change between iterations (it only changes when a new spectrum is input).

The previous neuron state $\mathbf{x}(t)$ must be added to the optically calculated vector $\mathbf{a} - \mathbf{Tx}$. This can be done on the output detectors with electronics. It can also be achieved by using $\mathbf{I} - \mathbf{T}$ for M_1 which now requires negative number encoding, such as space multiplexing, and the same external detector electronics. The outputs of the operational amplifiers are limited to lie between 0 and 1 (in the appropriate units). If the net has not converged, this vector is fed back to LED_1 for the next iteration. The arrangement (without negative number encoding and with a bias ηA added electronically to the amplifiers) is attractive because it allows us to control η by controlling the gain of the LEDs in P_1 and P_4 and the output bias. Adapting η during the iteration process is useful since it allows us to speed up the convergence of the neural net as is explained in Sec. IV.

One obvious alternative to our neural algorithm has to be considered. Since we express our problem as the minimization of a quadratic energy function, the following is a plausible alternative method: Write the quadratic energy function as

$$E = \mathbf{x}'\mathbf{A}\mathbf{x} + \mathbf{c}'\mathbf{x}. \tag{10}$$

To obtain the minimum of E analytically, differentiate (10) with respect to **x** and set the result equal to zero. This gives

$$\mathbf{x} = -(\tfrac{1}{2})\mathbf{A}^{-1}\mathbf{c}. \tag{11}$$

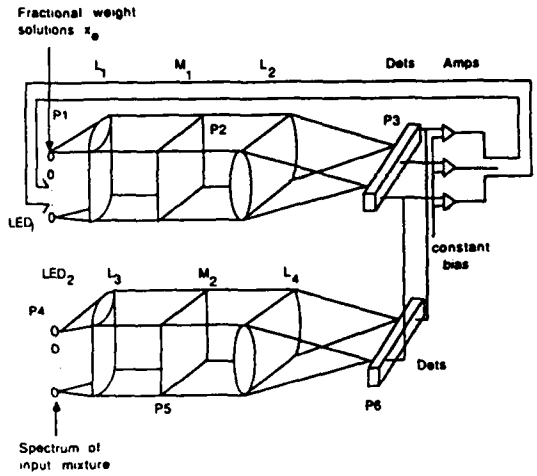


Fig. 2. Optical neural net architecture for the minimization procedure.

(It is assumed, without loss of generality, that A is symmetric). The problem with this approach is that the solutions do not satisfy the constraint that the x_e should lie in the range [0,1]. (The constraint that the fractions should sum to 1 can be enforced by rescaling the result obtained.) No simple analytic way of enforcing the range constraint exists; therefore our current method is preferred.

IV. Simulation Results

The neural net of Fig. 2 and the algorithm in Eq. (9) were simulated for the case of $N = 826$ wavelength samples and $E = 10$ elements. The minerals used were ten of the most common minerals (Table I). Their reflectance spectra contained samples at 1-nm intervals from 400–799 nm, and samples at 4-nm intervals from 800–1500 nm. The measurements were supplied by the Jet Propulsion Laboratory. A few examples of the reflectance spectra used (indicating the percentage of light that is reflected at the specified wavelength) are shown in Fig. 3. The mixture included in Fig. 3 consisted of 50% montmorillonite and 50% dolomite. The spectra of the mixtures were generated from the spectra of the minerals, using the linear model described in Sec. II. We also used spectra for different grain sizes of the minerals, since the grain size affects the reflectance spectrum. We refer to these as large (>125 μm) and small (<45 μm) grain sizes.

Table I. Minerals and Mixtures Used in the Simulations

Common minerals used	
Kaolinite	Illite
Alunite	Jarosite
Gypsum	Chalcedony (a quartz)
Montmorillonite	Chlorite
Calcite	Dolomite

Mixtures used
Dolomite/montmorillonite
Gypsum/dolomite/calcite
(Various percentage compositions of both)

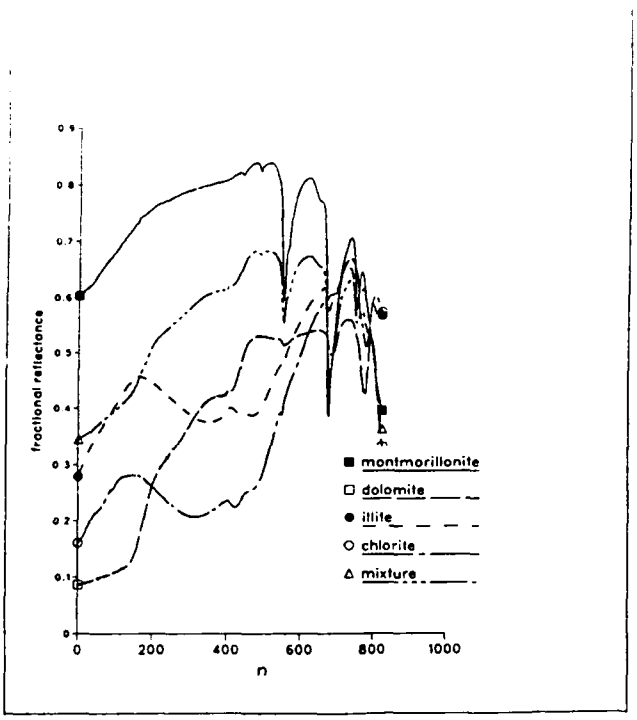


Fig. 3. Typical spectra of four minerals, and a mixture of two of the minerals.

Three sets of experiments were performed. In the first set, we investigated the ability of the neural net to determine the identity of an input spectrum that was one of the ten minerals (with only large, only small, and with both small and large grain sizes used). In the first case, the spectra K_n^e used to calculate the connection matrix T_{ef} in Eq. (6) belonged to samples with large grain sizes, whereas all the spectra used in the second case were obtained from small grain sizes, and the third case used some samples with large grain sizes and others with small grain sizes. Each of these 3 tests in the first set of experiments consisted of 100 runs. A different 1 of the 10 original spectra k^e was used as input spectrum c for 10 of the runs. The 10 runs with the same input spectrum differed from one another only in the different initial neuron conditions that were used, as we describe below. The inputs in this first set of experiments are referred to as pure inputs (i.e., only one mineral was present).

The second set of experiments investigated the ability of our algorithm to recognize the spectra of the pure large grain-size minerals in the presence of noise. Each spectral measurement was perturbed by $n\%$ of noise. This was achieved by adding a uniformly distributed random number to each measurement. This produced a random variation in the value of the reflectance by at most $n\%$ of its original value at each λ_n. Values used for n were 2.5% and 5% with 100 experimental runs executed for each noise value, as above.

Finally, we studied the performance of the neural net when mixtures were used as input without noise and with the same two nonzero noise levels used above present (2.5% and 5%). For each noise level, 10 different mixtures were used as input. These mixtures consisted of different fractional compositions of the elements of the 2 mixtures listed in Table I. For each noise level, 100 runs were again performed (10 input mixture spectra times 10 initial neuron conditions).

For each experimental run, we proceeded as follow: the 10×10 connection matrix T in Eq. (6) was calculated using the 10 spectra of the pure minerals, as described above. We set $A = 1.0$ to weigh the two terms in Eq. (3) equally, since both terms are of approximately equal importance to us. Ten neurons were used in each experiment, because 10 minerals were used. Initially, a random number uniformly distributed between 0 and $2/E = 0.2$ was assigned to each neuron. This assures that the expected value of the sum of the neuron activities equals 1, as it should be because the neuron activities represent fractional compositions. The 10-element vector a was then calculated using Eq. (7). This vector depends on the input spectrum to be identified. The step size parameter η was initially chosen equal to 1/Trace(T), since this is the largest value of η which is certain to lead to convergence. As the net converged to its stable state, convergence was accelerated (to reduce the number of iterations) by doubling η whenever the sum of the squares of the differences of the neuron activities at successive time steps decreased by a factor of 4. Since the change in neuron activity $\mathbf{x}(t+1) - \mathbf{x}(t)$ is linear in η from Eq. (4), the sum of the squares of this difference will be proportional to η^2. Thus, η should be scaled in proportion to the square root of the sum-of-squares. Changing η when the sum of the squared differences was a factor of four was used, since this choice gave good performance. Whenever the errors increased from one iteration to the next, we divided η by 1.5, where the 1.5 empirical factor has been found to be suitable for a large range of data. This assures that the system will remain stable by reducing η when it becomes too large for stable convergence.

The net was iterated according to Eq. (9) until it converged (that is, until no neuron activity x_e changed by more than a prespecified amount). For the pure inputs, the terminating tolerance was always chosen to be 10^{-4}, since this leads to reasonable accuracy (<6%) in most cases, without requiring an excessive number of iterations. For the mixture inputs a smaller terminating tolerance (10^{-5}) was required, because the correct answers are now not simply zero or one. These tolerances mean that the difference in the value calculated for any element x_e on successive iterations t and $t+1$ was less than 10^{-4} or 10^{-5}.

Our stopping criterion used small differences in the neuron states x between iterations. These are used to determine digitally when we enter the minimum-E region of the E vs x curve. With a lower processor accuracy (such as we would expect with an analog optical neural net), the processor accuracy is also the accuracy to which we can calculate each of the x-values. We should thus be able to obtain (for a 1% accurate processor) a final x-state within 1% of the energy minimum. This issue merits further research.

For the specific imaging spectrometer least-squares problem, monitoring the change in E rather than the change in $\mathbf{x}$ between iterations would also provide a useful stopping criterion. Our choice is more general, however, since it will provide a useful stopping criterion even if the energy magnitude of the best possible solution is not known.

Table II shows the results obtained. Column 1 describes the type of spectra K_n^e used for a given run. The first three tests (experimental set 1) used one input mineral with no noise present. The next two tests also used one input mineral, but with noise added (set 2). The final three tests involved mixtures as input, with and without noise (set 3). Column 2 lists the percentage of noise n by which the spectra were perturbed. Column 3 lists the number of iterations required to reach the stopping criterion in each case, and column 4 gives the precision of the result (the largest amount by which any stable neuron state differed from the true compositional fraction). Note that this is a worst case precision error.

We see that the neural net was successful in classifying the input spectra to reasonable accuracy. The worst performance occurred when small grain sizes were used. This occurred because the shape of the spectra of several of the minerals were very similar (see, for example, the spectra of illite and chlorite in Fig. 3). When the spectrum of only one of them (illite) was present, the net converged to a state containing a mixture of these two minerals (see test 2 in Table II). In this case, we decreased the terminating tolerance and found that after 5000 iterations the net was still slowly converging towards the correct solution. For such minerals, it is preferable initially to classify both into one class and then use postprocessing (e.g., using a smaller neural net with only those minerals detected in the first pass represented by neurons) to determine which was actually present. Alternatively, a representation more sensitive to the fine structure of the spectra (such as the derivative of the spectrum with respect to wavelength or the use of only a few wavelengths) can be used to discriminate such similar spectra. In all other cases the maximum error averaged over 100 runs per test was less than 5.5%. In the mixture results better precision was obtained at the cost of an increased number of iterations by decreasing the terminating tolerance. The results in Table II also indicate good performance in the presence of additive noise. When 2.5% noise was added to the spectra, no noticeable degradation in the precision occurred, but the net required more iterations to converge. Increasing the noise to 5% did not appreciably affect the performance of the net.

Table II. Simulation Results for the Determination of the Composition of an Input Element or Mixture

Input spectra	Noise percentage n (%)	Average number of iterations	Worst case error in any x_r (%)
Pure, large	0	738	3.39
Pure, small	0	1094	11.33
Pure, large and small	0	940	5.42
Pure, large	2.5	1155	5.16
Pure, large	5	1153	4.91
Mixture, large		2236	0.98
Mixture, large	2.5	3603	1.46
Mixture, large	5	3537	1.86

V. Summary and Conclusion

A new optical neural net architecture and algorithm were introduced to find the composition of a mixture given its spectrum and the spectrum of the possible minerals. A quadratic cost function is minimized to find the optimal composition. This includes the constraint that all fractions sum to unity. The constraint that all compositional fractions lie between zero and one is enforced by using neurons with a nonlinear transfer function.

Simulation results were presented that indicate that the neural net algorithm performs satisfactorily. Difficulties are due to input spectra that are very similar in shape. Techniques to overcome this were addressed. In general, the number of iterations required for the net to converge was large. This indicates that serial simulations of the net are not realistic for large applications, and emphasizes the necessity of using a parallel system such as our optical architecture.

This research was funded by a contract from the Strategic Defense Initiative Office of Innovative Science and Technology, monitored by the Office of Naval Research (Contract N00014-86-K-0599), with partial support from NASA Jet Propulsion Laboratory (Grant 958097).

References

1. J. Adams, P. Johnson, M. Smith, and S. Taylor-George, "A Semi-Empirical Method for Analysis of the Reflectance Spectra of Binary Mineral Mixtures," J. of Geophys. Res. 88, 3557–3561 (1983).
2. W. Lawton and M. Martin, "The Advanced Mixture Program—Principles and Algorithms," Technical Report IOM 384 (Jet Propulsion Laboratory, 1985).
3. R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine 2, 4–22 (1987).
4. G. A. Carpenter and S. Grossberg, Eds., "Neural Networks: Introduction to the 1 December Issue of *Applied Optics*," Appl. Opt. 26, 4909–4992 (1987).
5. D. Psaltis and N. Farhat, "Optical Information Processing Based on an Associative-Memory Model of Neural Nets with Thresholding and Feedback," Opt. Lett. 10, 98–100 (1985).
6. R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis* (Wiley, New York, 1973).
7. John J. Hopfield and D. W. Tank, "Computing with Neural Circuits: A Model," Science 233, 625–633 (1986).

CHAPTER 5

"Multitarget Tracking with Cubic Energy Optical Neural Nets"

Multitarget tracking with cubic energy optical neural nets

Etienne Barnard and David P. Casasent

A neural net processor and its optical realization are described for a multitarget tracking application. A cubic energy function results and a new optical neural processor is required. Initial simulation data are presented.

I. Introduction

Considerable interest currently exists in neural networks[1,2] due to their adaptive properties, fault tolerance, and high computational throughput. One can distinguish current neural processors by whether they concern pattern recognition and associative memories[3–5] or multivariate optimization.[6,7] Our concern is with the application of neural networks in optimization problems. As a specific case study, we consider multitarget tracking.

In Sec. II, we briefly review the evolution equations as used in neural minimization. Section III contains a definition of the specific problem we consider, and Sec. IV is a formulation of the constraints in our multitarget tracking problem as an energy function to be minimized. New optical architectures for the implementation of the equations in Sec. IV are then described (Sec. V). Simulation results are presented in Sec. VI. Our work contains three new ideas: the application of the Hopfield model to a multitarget tracking problem; the use of a nonquadratic energy function in the minimization problem and an optical architecture which can calculate the evolution of a system with such a nonquadratic energy function.

II. Neural Model

We use the Hopfield model[6] as a minimization network. We represent the state of the neurons by $X_i(t)$, where t is a time variable and i labels a particular neuron within the set of neurons. For an optimization problem, we wish to find the set of X_i that minimizes an energy function E, which is a function of the neural activities X_i. In this model, the evolution of the activity of each neuron (its rate of change with time) is described by

$$\frac{dX_i}{dt} = -\frac{\partial E}{\partial X_i}. \tag{1}$$

The time evolution of the energy function is

$$\frac{dE}{dt} = \sum_i \frac{\partial E}{\partial X_i}\frac{dX_i}{dt}. \tag{2}$$

To show that the model in Eq. (1) minimizes E, we substitute Eq. (1) into Eq. (2) and obtain

$$\frac{dE}{dt} = -\sum_i \left(\frac{\partial E}{\partial X_i}\right)^2. \tag{3}$$

Equation (3) shows that E is a decreasing function of time t. The energy E will converge to a local minimum as t progresses. Thus, the set of neural activities $\{X_i\}$ in the final stationary state describes a minimum energy state of the system.

In our work, this basic algorithm is modified by using discrete time[8] and by employing binary neuron activities X_i. With binary X_i, the neuron activity (neuron state) in Eq. (1) can now be replaced by

$$X_i = \begin{cases} 1 \text{ if } \dfrac{\partial E}{\partial X_i} < 0, \\ 0 \text{ if } \dfrac{\partial E}{\partial X_i} > 0, \end{cases} \tag{4}$$

that is, the state of neuron i is binary and depends on the energy as noted. The choice in Eq. (4) insures that the energy function is approximately minimized in the stationary state, as we now show.

With unit time steps, we replace dX/dt by ΔX and dE/dt by ΔE. To find the change in energy due to a state change of neuron i, we recall the Taylor expansion of $E(\{X_i + \Delta X_i\})$ in the vicinity of $E(\{X_i\})$:

$$E(\{X_i + \Delta X_i\}) = E(\{X_i\}) + \sum_i \frac{\partial E}{\partial X_i}\Delta X_i + \sum_i \sum_j \frac{\partial^2 E}{\partial X_i \partial X_j}\Delta X_i \Delta X_j + \ldots. \tag{5}$$

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 18 February 1988.
0003-6935/89/040791-08$02.00/0.

One can therefore calculate the change in neural energy ΔE due to the state changes ΔX_i of the neurons by $\Delta E = E(\{X_i + \Delta X_i\}) - E(\{X_i\})$. From Eq. (5), keeping only the first term,

$$\Delta E = \sum_i \frac{\partial E}{\partial X_i} \Delta X_i + \ldots. \tag{6}$$

Using Eq. (4) in Eq. (6), we see that, if $\partial E/\partial X_i$ is negative, we set the state X_i of neuron i to one and if $\partial E/\partial X_i$ is positive, we set X_i to zero. Thus, ΔE is negative. In Eq. (6) the higher-order terms were omitted. Thus, the prior analysis is only approximately correct, and the energy of the system actually can increase on some iterations. It turns out that this is more beneficial than harmful, since it allows the system to escape from shallow local minima.

Since we express the multitarget tracking problem as a constrained optimization problem, it is also possible to use conventional (non-neural) optimization techniques. However, such techniques are not suitable for optical implementation, and generally require much more computation than the Hopfield net. We therefore restrict our attention to the techniques described in this section.

III. Problem Definition and Case Study

The minimization problem we consider is a multitarget tracking problem. The scenario we consider assumes:

(1) N_T targets are to be tracked with N_T known and fixed (being determined by the track initiator).

(2) There are N_M measurements each time step or frame of data. N_M is fixed and is the maximum number of measurements we will accept in any time frame. This is achieved as explained below. If the number of peaks (measurements) is less than N_M, we lower the detection threshold or insert artificial measurements to insure that at each time we have $N_M \geq N_T$ measurements.

(3) The targets do not accelerate appreciably during the time steps under investigation and thus their trajectories are approximately straight lines.

(4) Each target corresponds to no more than one measurement at each time.

(5) Each measurement is due to no more than one target at each time. (That is, we ignore crossing targets for now.)

(6) At each time step, each target must be assigned to one measurement. Our selection of N_M in item (2) insures that $N_M \geq N_T$ so that this rule can be satisfied.

The optimization problem is to assign one track to each target, i.e., for each time step one set of detected objective is to find the N_T best straight lines in the given data.

IV. Problem Formulation

We first present our notation, introduce the distance measures we wish to minimize, and then develop a neuron energy description. We denote the measured position vectors at time steps a, $a+1$, and $a+2$ by $\mathbf{r}_\alpha^a$, $\mathbf{r}_\beta^{a+1}$, and $\mathbf{r}_\gamma^{a+2}$, respectively. The subscripts α, β, and γ are used to refer to a particular one of the N_M different measurements at a given time step, the time steps being indexed by the superscript. We denote the vector difference between a specific measurement (one of the α) at time step a and one of the β measurements at time step $a+1$ by $\mathbf{d}_{\alpha\beta}^a = \mathbf{r}_\alpha^a - \mathbf{r}_\beta^{a+1}$. Similarly, $\mathbf{d}_{\beta\gamma}^{a+1}$ denotes the vector distance between a measurement β at time step $a+1$ and a measurement γ at $a+2$. In terms of these vector distances, the vector distance measure we wish to minimize for a sequence of three time steps for all measurements is

$$D_{\alpha\beta\gamma}^a = \|\mathbf{d}_{\alpha\beta}^a - \mathbf{d}_{\beta\gamma}^{a+1}\|. \tag{7}$$

The minimum of Eq. (7) assigns one measurement in each of time steps a, $a+1$, and $a+2$ to the same target. Note that D in Eq. (7) is the norm of a vector difference. This ensures that two successive distance vectors (for time steps a and $a+1$) should be collinear to minimize D; that is, D is minimized for straight line tracks. With equal time step increments and a straight line trajectory with no acceleration, the two distance vectors will be equal (for true target measurements). Thus D will be 0 for the case of three collinear and evenly spaced measurements in three successive frames.

The measure D will now be used as a basis for the description of an energy function E which, when minimized, solves our problem. We label each binary neuron with three indices, such as $X_{i\alpha a}$, where i is the target index, α is the measurement index, and a is the time step index. This neuron is active (i.e., $X_{i\alpha a} = 1$) if the ith target is associated with a specific position vector $\mathbf{r}_\alpha^a$ (one of the measurements α) at time step a, and otherwise $X_{i\alpha a} = 0$. The energy function to be minimized for the optimization problem in Sec. III can be written as

$$\begin{aligned} E = {} & A_1 \sum_a \sum_\alpha \sum_i \sum_{j \neq i} X_{i\alpha a} X_{j\alpha a} \\ & + A_2 \sum_i \sum_a \sum_\alpha \sum_{\beta \neq \alpha} X_{i\alpha a} X_{i\beta a} \\ & + A_3 \sum \left(\sum_i \sum_\alpha X_{i\alpha a} - N_T \right)^2 \\ & + A_4 \sum_a \sum_i \sum_\alpha \sum_\beta \sum_\gamma D_{\alpha\beta\gamma}^a X_{i\alpha a} X_{i\beta(a+1)} X_{i\gamma(a+2)}, \end{aligned} \tag{8}$$

where A_1–A_4 are positive constants. Their choice is discussed in Sec. VI.

We now discuss the terms in this energy function to provide an understanding of it. We first note that all terms are positive semidefinite. Consider the first term: each term in this sum is either 0 or 1 (since binary neurons are employed). Note that $X_{i\alpha a}$ and $X_{j\alpha a}$ denote neuron states associated with targets i and j (any of the N_T targets) and some measurement α (of the N_M) at time step a. The first term contains the sum over the measurements and time steps of products of these neurons. Since only the target index (i or j)

differs in the $X_{i\alpha a}X_{j\alpha a}$ product, a given term in term 1 can be one only if targets i and j are assigned to the same measurement α at the same time step a. Since we sum over α, a, i, and j, term 1 is zero if and only if at each time step no measurement is assigned to more than one target. Thus, minimization of this first term occurs when each measurement is associated with no more than one target. It therefore enforces condition (5) in Sec. III.

The second term in Eq. (8) consists of a sum of products with different measurement indices (α and β) on the neurons. It is therefore minimized when one target is associated with no more than one measurement (α or β) at the same time step a. Thus term 2 is included so that the system satisfies condition (4) in Sec. III.

We next consider term 3. For a fixed time a, the set of neurons $X_{i\alpha a}$ for the various target indices i and measurement indices α can be arranged in a matrix with horizontal index α and vertical index i. When one measurement has been assigned to each target, this matrix has a single one in each row i, indicating which measurement is assigned to this target. Thus, $\sum_i \sum_\alpha X_{i\alpha a}$ for a fixed time a is the number of nonzero entries of the matrix above. This sum equals N_T when condition (5) of Sec. III is satisfied. Thus, term 3 is minimized when all N_T targets are each associated with one measurement at each time step; it is included so that condition (6) in Sec. III is satisfied. Hence, the first three terms in Eq. (8) ensure that the measurement–target matching is admissible.

In term 4, both the time step and measurement indices on the three neurons in the product differ. This term selects a measurement (from each of the sets labeled by α, β, and γ) in each of three successive time frames for each target. The measurement–target pairs are selected such that these three measurements lie closest to a straight line, with the search done for each target and for each measurement α in each frame. To see how this is accomplished, recall that $D^a_{\alpha\beta\gamma}$ in Eq. (7) is calculated for three successive time steps. The three neurons in term 4 in Eq. (8) have their time indices appropriately stepped. For a fixed time frame, the three X terms can each be represented by a matrix with horizontal index α and vertical index i, as before. For a fixed target i, the neuron choices to be considered occur in the same row (row i) in each of these matrices. Each row of each matrix should have only a single one, because of condition (5) in Sec. III. The best choice for the position of these ones is determined as follows.

Consider that there are ten measurements in each frame. We select a measurement α in frame a. For the single measurement α chosen, there are ten possible choices for the measurement (indexed by β) in the second frame and for each of these there are ten possibilities for the third measurement indexed by γ. For each of these 100 combinations, the three X factors in term 4 could all be 1, but for only one set of these will D be small. Minimization of E for this term ensures that the set of three successive measurements chosen (for each measurement in the first frame) will be the set closest to a straight line. The summation over a implies that this minimization is repeated for each time step (using the prior two frames of data).

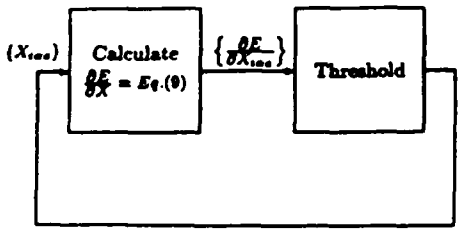


Fig. 1. Block diagram of the multitarget tracking neural processor.

A possible variation to the energy function in Eq. (8) is the omission of the first term. Then, one would not be enforcing the assignment of only one target to each input measurement (this case arises if two paths of different tracks cross). In our simulations, this term was retained. We intend to do more work on the target-crossing problem in the future.

Note that the energy function in Eq. (8) enables us to tolerate both spurious measurements and the absence of measurements for some targets at some time steps. To achieve this, let N_M be the largest number of measurements at any of the N_P time steps. If a given frame has $M < N_M$ target measurements, we set $N_M - M$ measurements equal to the zero vector. (In our reference frame the zero vector lies in the center. This choice minimizes the effect of missing measurements on D, for the case of a uniform spatial distribution of targets). Spurious measurements (if they have random position vectors, as they should) will not be assigned to true target tracks because of the energy minimization step. If both spurious and missing measurements are present, we have found that the spurious measurements are assigned to the same tracks as the missing measurements (zero vectors).

The time evolution of the neurons is required to result in a neural system that minimizes E in Eq. (8). This is given by the derivative of Eq. (8), i.e.,

$$\frac{\partial E}{\partial X_{i\alpha a}} = 2A_1 \sum_{j\neq i} X_{j\alpha a} + 2A_2 \sum_{\beta\neq\alpha} X_{i\beta a} + 2A_3\left(\sum_j \sum_\beta X_{j\beta a} - N_T\right) + A_4 \sum_\beta \sum_\gamma \left[D^a_{\alpha\beta\gamma} X_{i\beta(a+1)} X_{i\gamma(a+2)} + D^{a-1}_{\beta\alpha\gamma} X_{i\beta(a-1)} X_{i\gamma(a+1)} + D^{a-2}_{\beta\gamma\alpha} X_{i\beta(a-2)} X_{i\gamma(a-1)}\right]. \quad (9)$$

Figure 1 shows the block diagram of the neural multitarget tracker described by Eqs. (4), (8), and (9). We produce $\partial E/\partial X$ in Eq. (9) from X and threshold $\partial E/\partial X$ as defined in Eq. (4) to produce the new X with E given by Eq. (8) from which $\partial E/\partial X$ is obtained in a closed loop. In this design, the activities of the neurons evolve according to Eqs. (9) and (4), and thus their states will evolve to a steady-state energy minimum. This minimum indicates which target should be associated with which measurement at each time step.

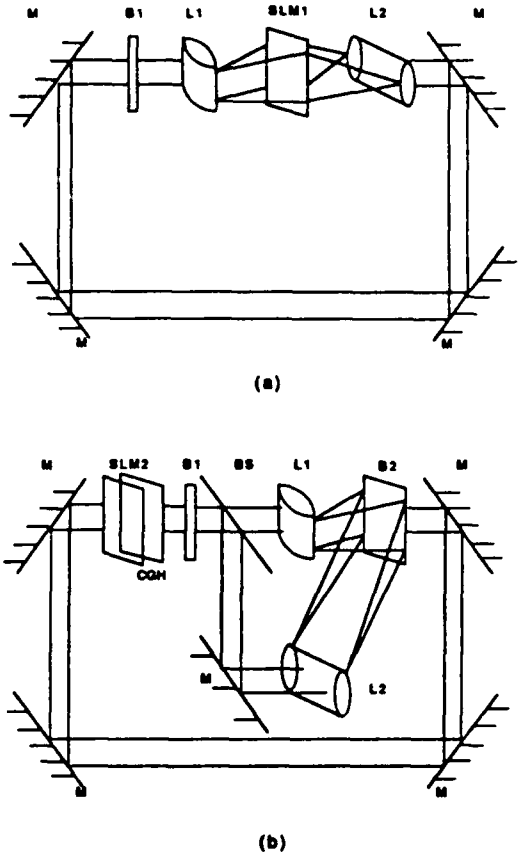


Fig. 2. Schematic of an optical neural processor with (a) quadratic energy terms and (b) cubic energy terms.

We now discuss how to realize the terms in Eq. (9) as linear algebra functions. We consider the case of N_T = 10 targets, N_M = 10 measurements, and N_P = 3 time steps. There are $N_T \times N_M \times N_P$ = 300 neurons that represent the different $X_{i\alpha a}$. We represent these neurons as a vector $\mathbf{x}$ with elements x_k, where each value of the index k denotes a different (i,α,a) combination. In steady state, $\mathbf{x}$ will have thirty "one" entries (ten measurement–target pairs for each of three time step values a). Term 1 in Eq. (9) is the sum of a number of such vectors and can thus be described as a matrix–vector product $2A_1\mathbf{T}_1\mathbf{x} = \mathbf{y}_1$. The elements T_{kl} of the binary connection matrix $\mathbf{T}_1$ are described by

$$T_{kl} = T_{i\alpha a,j\beta b} = \delta_{\alpha\beta}\delta_{ab}(1 - \delta_{ij}), \tag{10}$$

where the indices k and l denote the different sets of target/measurement/time parameters $(i\alpha a)$ and $(j\beta b)$, respectively. Since both k and l range over 300 values, $\mathbf{T}_1$ is a 300 × 300 matrix. This $\mathbf{y}_1$ term has nonzero contributions to the output only for indices corresponding to different targets $(i \neq j)$ but the same measurement $(\alpha = \beta)$ and time step $(a = b)$.

Terms 2 and 3 in Eq. (9) are other sums of vectors $\mathbf{x}$ and can likewise be written as matrix–vector products $2A_2\mathbf{T}_2\mathbf{x} = \mathbf{y}_2$ and $2A_3\mathbf{T}_3\mathbf{x} = \mathbf{y}_3$. For these first three terms, the connection matrices are fixed and thus we can form $\mathbf{Tx} = (2A_1\mathbf{T}_1 + 2A_2\mathbf{T}_2 + 2A_3\mathbf{T}_3)\mathbf{x}$ in a single step with $\mathbf{T}$ and the constants A_1–A_4 fixed for all problems.

Term 4 in Eq. (9) is more complex. Each of the three parts of this term is similar and contains the product of two different neuron states which we can relabel as X_k and X_l times a tensor $\mathbf{D}$ of rank three. If we think of X_k and X_l as components of a vector $\mathbf{x}$, the product of two different neuron states X_kX_l for all k and l is a matrix with components X_kX_l (where k and l are the row and column indices, respectively). This is the vector outer product (VOP) matrix $\mathbf{xx}^t$, where the superscript t denotes the transpose. In terms of this new vector labeling scheme, term 4 can be written as

$$y_{4j} = \sum_{k,l} D_{jkl}X_kX_l, \tag{11}$$

where y_{4j} denotes the jth component of term 4 in Eq. (9). We can view $\mathbf{D}$ as a number of matrices. The sum of products in Eq. (11) for a given j is the sum of the point-by-point products of the elements of the VOP matrix and one of the matrices in $\mathbf{D}$. The result for all j is a vector $\mathbf{y}_4$, which is called[9] the tensor–matrix inner product, i.e.,

$$\mathbf{y}_4 = \mathbf{D} \cdot \mathbf{xx}^t. \tag{12}$$

Since this is not a simple matrix–vector product, it cannot be calculated in the same way as the other three terms in Eq. (9). In addition, $\mathbf{D}$ changes with the input data whereas the matrix $\mathbf{T}$ in the other terms is fixed. We now investigate how the linear algebra operations and thresholding described in this section can be done optically.

V. Optical Architecture

As has often been noted, connectionist architectures are well suited for optical implementation since optical systems easily achieve large numbers of interconnects.[10] In the optical design of the Hopfield net by Psaltis and Farhat,[11,12] a matrix–vector multiplier was sufficient. This optical realization is suitable for implementing time evolution equations that are linear in the neuron activities X (or equivalently, neural systems and applications in which the energy function is quadratic in X). Such an optical architecture is most efficient if only non-negative connection matrices are involved. Thus, the first two terms in Eq. (9) and the positive definite part $(2A_3 \sum_j \sum_\beta X_{j\beta a})$ of the third term can easily be calculated by unipolar optical matrix–vector multiplications. Our optimization problem involves one energy term which is cubic (term 4) and a negative term (part of term 3). The optical realization of these terms is now discussed.

Equation (9) can be realized on the optical system of Fig. 2 as we now detail. This optical system is best drawn in two parts: Fig. 2(a) (which performs a matrix–vector multiplication and implements terms 1–3) and Fig. 2(b) (which implements term 4). The data plane $B1$ is common to both parts of the optical system. For simplicity, only the essential lenses are shown. In Fig. 2(a), the vector data on a 1-D bistable device[13] $B1$ is the current neuron state $\mathbf{x}$. It is imaged vertically

and expanded horizontally by $L1$ onto a 2-D spatial light modulator (SLM1) which contains the matrix interconnection data **T** in Eq. (10). This is a fixed interconnection pattern that is independent of the data. Thus, it can be recorded on film and need not be altered. The light leaving SLM1 represents the point-by point products $T_{ij}x_j$. This light is integrated vertically and input back to $B1$ (by four mirrors M in the version shown). This forms the matrix–vector product $\mathbf{Tx} = (\sum_j T_{1j}x_j, \sum T_{2j}x_j, \ldots, \sum_j T_{nj}x_j)$.

The same $B1$ is also present in the system of Fig. 2(b). In Fig. 2(b), its output is expanded horizontally by $L1$ and rotated by 90° and expanded vertically [by the beam splitter (BS), mirror (M), and lens $L2$]. These two expanded patterns are superimposed on a second bistable device $B2$ with horizontal and vertical indices k and l. The light intensity incident on element (k,l) of $B2$ is $X_k + X_l$. When the threshold for $B2$ is set to trigger only if both inputs are active, the $B2$ output is the binary VOP of the $B1$ data. This VOP matrix is imaged onto SLM2 where it multiplies several multiplexed D matrices element by element. The computer-generated hologram (CGH) behind SLM2 directs the proper element-by-element products to different portions of $B1$. To implement term 4 on this system, we form **x** on $B1$, the VOP matrix $\mathbf{xx}^t$ on $B2$, and the tensor–matrix inner product $\mathbf{y}_4 = \mathbf{D} \bullet \mathbf{xx}^t$ on $B1$ using SLM2 and the CGH. We now consider the multiplexing data format on SLM2 and the CGH used.

In our simplified index notation (Sec. IV), the elements of the $\mathbf{y}_4$ vector output due to term 4 are given by Eq. (11). Consider the case when j, k, and l range from 1 to 3 in D_{jkl}, X_k, and X_l. The neuron vector **x** then has three elements and the VOP matrix on $B2$ is 3×3. Each element X_kX_l must multiply the three different elements D_{jkl} corresponding to the three different values that j can take given the indices k and l. One possible multiplexed arrangement for the SLM2 data D_{jkl} is shown in Fig. 3, with the VOP elements in row 1 of $B2$ corresponding to $(k,l) = (1,1), (1,2), (1,3)$ and the elements of row 2 corresponding to $(k,l) = (2,1), (2,2), (2,3)$, etc. The spatial size of the elements on $B2$ and SLM2 and the imaging optics (not shown) from $B2$ to SLM2 are such that VOP element (1,1) illuminates the first three elements in column 1 of Fig. 3 (i.e., D_{111}, D_{211}, and D_{311}), VOP element (1,2) corresponding to X_1X_2 illuminates the first three elements of column 2 (D_{112}, D_{212}, and D_{312}), etc. The bold lines in Fig. 3 indicate regions of SLM2 illuminated by one element of $B2$. Since **D** is a tensor of rank 3, it is not possible to assign one spatial dimension (horizontal or vertical) to each rank (as is possible with a tensor of rank 2, i.e., a matrix). This arrangement in Fig. 3 multiplies each VOP element X_kX_l by the three possible j values in D_{jkl} and thus forms the point-by-point product of the VOP matrix and the different D matrices. The CGH behind SLM2 focuses all products with the same j onto the same region of $B1$ (i.e., for the 3×9 example in Fig. 3, it sums the light leaving the first, fourth, and seventh rows, the light leaving the second, fifth, and eighth rows, etc). This forms the sum over k and l of $D_{jkl}X_kX_l$ for each j in a different region of $B1$. A CGH could be placed between $B2$ and SLM2 to replicate the $B2$ data onto the proper regions of SLM2 such that the CGH behind each region (3×3 region for the example in Fig. 3) of SLM2 could be a simple spherical lens plus a grating at the required spatial frequency and orientation. However, since the CGH is fixed and independent of the input data, it appears that it can be fabricated on film with sufficient resolution to allow one CGH to be used with improved light budget efficiency. We detail this later.

D_{111}	D_{112}	D_{113}
D_{211}	D_{212}	D_{213}
D_{311}	D_{312}	D_{313}
D_{121}	D_{122}	D_{123}
D_{221}	D_{222}	D_{223}
D_{321}	D_{322}	D_{323}
D_{131}	D_{132}	D_{133}
D_{231}	D_{232}	D_{233}
D_{331}	D_{332}	D_{333}

Fig. 3. Details of the values written on SLM2.

Next, we consider how the negative part of term 3 in Eq. (9) is handled. Recall that $B1$ is common to both parts of the system [Figs. 2(a) and (b)]. Thus, the input to $B1$ contains the sum of all the non-negative terms in $\partial E/\partial X_{i\alpha a}$ in Eq. (9), i.e., the jth element on the input side of $B1$ is $\partial E/\partial X_j + 2A_3N_T$ [where j corresponds to $(i\alpha a)$]. Thus, we set the threshold of $B1$ to be $2A_3N_T$ and hence achieve the subtraction of the positive and constant $2A_3N_T$ portions of term 3 by thresholding without the need to compute negative numbers and the proper neuron vector **x** emerges from $B1$. We note that the contents of SLM2 need not change between iterations (in minimizing the energy E) for a given set of input measurements. Its contents change for each set of input measurements, but such distance calculations are needed for most multitarget tracking problems.

We next consider an improved version of the system of Fig. 2(b) with reduced space–bandwidth product for $B2$ and SLM2. To see the significance of this, consider the case of $N_T = 6$, $N_M = 7$, and $N_P = 5$. The vector **x** has $6 \times 7 \times 5 = 210$ components, the VOP has 210×210 components, and SLM2 requires $210 \times 210 \times 210$ pixels. Clearly, for large values of N_T, N_M, and N_P, this architecture becomes unrealistic. Fortunately, not all the terms in $\mathbf{xx}^t$ are required and most elements of **D** are zero. In Eq. (9) we see that only those values of $X_{i\alpha a}X_{j\beta b}$ with $i = j$ are used. To take advantage of

this, we divide the vector $\mathbf{x}$ into N_T smaller vectors $\mathbf{z}_i$ ($i = 1, \ldots N_T$), one for each target j, with each vector of size $N_M \times N_P$. Thus, the full vector $\mathbf{x}$ can be written as $\mathbf{x}^t = (\mathbf{z}_1^t, \mathbf{z}_2^t, \ldots, \mathbf{z}_{N_T}^t)$. This allows us to calculate the VOP $\mathbf{z}_i\mathbf{z}_i^t$ of each vector separately, multiply each by the proper elements of $\mathbf{D}$ point by point, and sum up the products as indicated in Eq. (9).

We now discuss how to efficiently separate the tensor $\mathbf{D}$ into several smaller matrices. Recall that the full tensor $\mathbf{D}$ is described by $N_TN_MN_P$ matrices, each of dimension $N_TN_MN_P$. However, the only nonzero elements of these matrices occur for $i = j$ and do not depend on the value of i or j (the entries of $\mathbf{D}$ do not depend on the target, since the calculations of the elements of $\mathbf{D}$ involve only distance calculations on the measurements). The fact that only three adjacent time steps a are included in the $\mathbf{D}$ calculation further reduces the number of nonzero entries. We chose to separate the $\mathbf{D}$ tensor into N_M matrices, each of dimension N_MN_P. To see how this is possible, recall that each distance measure is associated with three measurements (α, β, and γ) at three different sequential time steps. A given set of pairs of two measurements at two different (not necessarily successive) time steps is described by a matrix of dimension N_MN_P. There are N_M possibilities for the third measurement in the other time step of the three in sequence. Thus there are N_M such matrices, each of dimension N_MN_P, that describe the tensor data $\mathbf{D}$.

This division is attractive since each of the $N_T = 6$ reduced size 35×35 (when $N_MN_P = 35$) VOP matrices can now be multiplied by each of the $N_M = 7$ D matrices. After summation of the proper point-by-point products, the output is $N_T = 6$ vectors, each of dimension $N_MN_P = 35$, i.e., the $6 \times 35 = 210$ element neuron state vector. Thus, this new arrangement requires that we calculate six 35×35 VOPs (i.e., $B2$ requires only $6 \times 35 \times 35 = 7350$ elements), and SLM2 is only of size $7 \times (35 \times 35)$. We have thus reduced the space–bandwidth product of SLM2 by a factor of over 1000. $B1$ is still a 1-D SLM of size $N_TN_MN_P = 210$ elements (one for each element of $\mathbf{x}$). This size for $\mathbf{x}$ is still much less than for cases when one assigns one neuron for each possible single target state (position in x and y) for every time step n (i.e., xyn neurons, where x and y are the number of pixels in the (x,y) projections of the measurement space, respectively). Our assignment of one neuron for each measurement for each target for each time results in a much smaller number (since the number of measurement points per frame is usually much less than the number of 2-D pixels in one image frame).

The N_T VOPs of the partitioned $B1$ data can be produced on $B2$, by replacing $L1$ in Fig. 2 by a CGH that is a set of N_T cylindrical lenses with gratings at different orientations and with different spatial frequencies. This $L1$ lenslet CGH array focuses the N_T sections of $\mathbf{x}$ on the vertical $B1$ device onto N_T different horizontal regions of $B2$ and replicates the vector data in each of the N_T sections horizontally at $B2$. As before, $L2$ expands $\mathbf{x}$ vertically onto $B2$. After thresholding, the result on $B2$ is the desired N_T VOPs of the N_MN_P element vector in each of the N_T regions on $B1$. These VOPs thus emerge from $B2$ as N_T matrices horizontally separated, with each matrix of dimension N_MN_P.

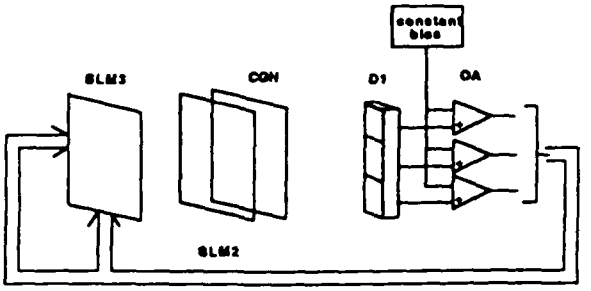


Fig. 4. Alternative optical architecture to implement cubic energy terms.

The architecture in Fig. 2 (even with the reduced SLM2 and $B2$ resolution requirements) requires fast nonlinear optical devices for $B1$ and $B2$ (since they limit the speed for one iteration of the system). Since such devices are not yet readily available, we present the alternative architecture of Fig. 4 that does not require a nonlinear optical device. Rather, it computes the vector outer product by feeding the $\mathbf{z}$ vectors to the rows and columns of an electroded SLM such as a ferroelectric liquid crystal.[14] Thus, SLM3 in Fig. 4 can be substituted for the 2-D bistable device $B2$ in Fig. 2(b). The outer products formed by SLM3 are imaged onto SLM2, where they multiply D, and the CGH focuses the terms belonging to the same sum to the same point on the detector array $D1$ which now replaces $B1$ in Fig. 2. The thresholding is done electronically by an array of operational amplifiers fed from the detectors $D1$. These detector outputs provide the electronic inputs to SLM3 in Fig. 4.

The architectures which we have introduced in this section are quite complicated. We do not find this surprising: it is well known that the multitarget tracking problem is very hard. Therefore, any parallel architecture which attempts to solve this problem in real time will probably be rather sophisticated.

VI. Simulation Results

The above neural net and algorithm were simulated for the case of $N_T = N_M = 4$ and $N_P = 5$. A $256 \times 256 \times 256$ 3-D (x,y,z) space was used. The initial four measurement positions were chosen from a random number generator. The velocities and directions of each target were similarly chosen. $N_P = 5$ equally spaced time steps were generated for each target. The target directions were evenly distributed over 360°. They generally ranged in length (in a 2-D projection) by a factor of 5:1 with the longest track of five time steps occupying ~70% of the field of view. To simulate imperfect measurements, each position was perturbed by n% of noise. This was achieved by adding a uniformly distributed random number to the measurement. This produced a random variation in the location of the measurement by at most n% of its distance

from the origin (the center of the 256 × 256 × 256 space).

For each run (corresponding to a given value n% of noise), ten different initial data conditions (initial target positions, velocities, and directions) were used. Thus, for each value of noise, ten sets of four target tracks were processed. The initial conditions (the neuron states at which the neurons were started in the processing) were chosen by randomly perturbing the all-equal condition (in which each coordinate is assigned to each target with equal probability). This is detailed more fully in Hopfield and Tank[6] and motivation for randomizing initial conditions is also provided there. Ten different random perturbations were used in the ten simulations in each run. The A_1–A_4 coefficients were chosen to equally weight the first three terms in Eq. (9), i.e., $A_1 = A_2 = A_3 = 15$ with A_4 chosen to be less ($A_4 = 1.8$). Larger A_1–A_3 values were used to give more weight to the first three terms in Eq. (9), i.e., we must have the proper form for the matrix X_{iaa}. A_1 and A_2 should be chosen equal (since these terms correspond to enforcing the correct row and column structure, respectively, and are thus equivalent by symmetry). A_3 could differ from these terms since it multiplies a different type of term; on the other hand, the first three items in Eq. (9) have similar roles and it was found that $A_3 = A_1 = A_2$ gave good results. Term 4 is given less weight ($A_4 = 1.8$) since it involves the sum of more products than do the other terms and satisfying conditions (1), (2), (4), (5), and (6) in Sec. III (terms 1–3) is essential. No detailed optimization of A_1–A_4 was attempted.

The coefficients $D^{a}_{a\beta\gamma}$ and the connection matrix (T) were calculated. The threshold $2A_3N_T$ in term 3 in Eq. (9) was slightly increased (from $2A_3N_T$ to $2A_3N_T$ + 0.035). We found this to be helpful when noise is present. Such an increase compensates for the neglected higher-order terms in the Taylor expansion in Eq. (6). In the simulation, the neural activities were updated as they were calculated, i.e., the state of the first neuron was calculated (using the most recent values of all other neurons) and then the state of neuron one was updated before calculating the state of neuron two, etc. This better models[8] a continuous time rather than a discrete time system. After one set of updates of the neurons, a new iteration commenced until the neural net converged. The serial mode neural updating results in faster convergence than if all new neuron states are calculated in parallel and simultaneously fed back, as is usually done.[7]

Table I. Simulation Results for Tracking of 4 Targets Through 5 Time Steps

Noise (%)	Successful (%)	Average no. of iterations
0	100	19
2.5	100	16
5	90	20
7.5	80	13

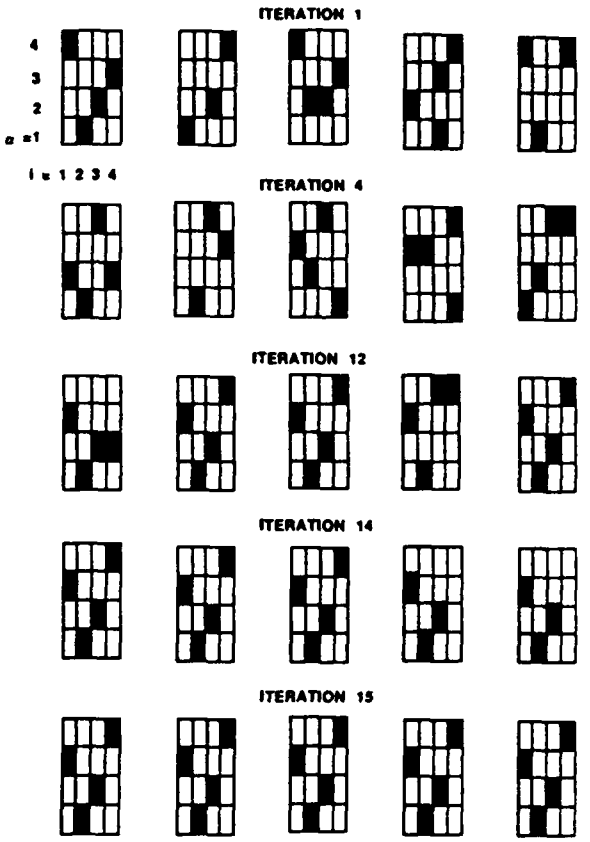


Fig. 5. Typical example of the time evolution of the neural states.

Table I shows the results obtained. Column 1 lists the percentage noise (positional variation) introduced into the measurements. Column 2 gives the percentage of runs that converged in less than 50 iterations and column 3 gives the average number of iterations to convergence for these cases. We restricted the number of iterations to 50. If convergence is not obtained after 50 iterations, we call this an error. If the system converged to the wrong set of measurement–target pair assignments, this is also an error. As seen, the neural net converged in much less than 50 iterations on the average. Also, we found that whenever the net converged, it converged to the correct target–measurement matching. A proof of this remains to be derived (but from these initial tests, one should be able to accept results that converge with a high probability). At low noise (n = 0% or 2.5%), the neural net converged to the correct solution for all 40 target tracks (i.e., excellent performance was obtained). As noise increased, correct convergence or two other conditions occurred (the neural net wandered with no apparent trend to convergence or it oscillated between two states, one being the correct solution and the other differing in one target–measurement pairing).

Figure 5 shows a representative example of the evolution of the neurons to a stable state for one set of data. In Fig. 5, each rectangle in the 5 × 5 grid shown represents the 4 × 4 matrix X_{iaa} for i = 1–4, α = 1–4, and a fixed (the assignments at one time step). The

matrices from left to right on one row correspond to different time steps a = 1–5 and the matrices in the different rows are the neuron states after various numbers of iterations, as indicated. The horizontal axis of each matrix has four divisions, corresponding to the four different targets which are assigned to the measurements. The four different measurements in a given time frame occupy the various rows of the indicated matrix. A dark spot at position (i,α) indicates that target i has been assigned to measurement α. The neural net has converged if the matrices are unchanged between two adjacent iterations. The targets are correctly tracked if $X_{i\alpha a}$ (for a fixed) has only one entry per row and one entry per column and row and column entries are the same in all time frames. This is because the measurements for this example were generated such that the same target was given the same number in all time frames.

From Fig. 5, the target–measurement associations appear random after the first iteration (i.e., the neural net has assigned many of the measurements to only one target and vice versa). They converge and eventually reach a correct solution after 15 iterations.

VII. Summary and Conclusion

A neural net energy minimization formulation of multitarget tracking with a reduced number of neurons was presented. Cubic energy terms were present and an optical architecture to implement this neural net was described. Initial simulations indicate that the algorithm has desirable performance, even in the presence of random noise. We expect improved performance when the optimal values for the A coefficients are understood, and when the characteristics of good initial neural states are known.

The formulation presented can be extended in a variety of ways. Other sensors than the simple position sensors we employed can be introduced and nonstraight tracks (i.e., accelerating targets) can be considered. These extensions will not complicate the energy formulation or optical architecture that we have presented, since they require only that the D tensor be calculated in a different manner.

The main reason conventional tracking algorithms are so slow is because they enforce a serial structure onto a process that is essentially parallel: the various targets are being measured simultaneously, but are processed one by one. The parallelism of our neural architecture means that it can perform at a much higher rate. We therefore believe that this architecture is prototypical of the types of system that will have to be used to successfully deal with a complicated multitarget scenario.

Funding for this research was provided by a contract from the Strategic Defense Initiative Office of Innovative Science & Technology, monitored by the Office of Naval Research (contract N00014-86-K-0599).

References

1. R. P. Lippmann, "An Introduction to Computing with Neural Nets," IEEE Acoust. Speech Signal Process. Mag. **2**, 4 (1987).
2. G. A. Carpenter and S. Grossberg, Eds., "Special Issue on Neural Nets," Appl. Opt. **26**, 4909–4992 (1987).
3. B. V. K. Vijaya Kumar and B. L. Montgomery, "A Direct Storage Nearest Neighbor Associative-Memory Model for Optical Pattern Recognition," Appl. Opt. **25**, 3759 (1986).
4. G. A. Carpenter and S. Grossberg, "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine," Comput. Vision Graphics Image Process. **37**, 54 (1987).
5. K. Fukushima, S. Miyake, and T. Ito, "Neocognitron: a Neural Network Model for a Mechanism of Visual Pattern Recognition," IEEE Trans. Syst. Man Cybern. **SMC-13**, 826 (1983).
6. J. J. Hopfield and D. W. Tank, "Computing with Neural Circuits: a Model," Science **233**, 625 (1986).
7. M. Takeda and J. W. Goodman, "Neural Networks for Computation: Number Representations and Programming Complexity," Appl. Opt. **25**, 3033 (1986).
8. J. J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities," Proc. Natl. Acad. Sci. U.S.A. **79**, 2554 (1982).
9. M. R. Spiegel, *Vector Analysis* (Schaum, New York, 1959).
10. L. D. Hutcheson, Ed., "Special Issue on Optical Interconnections," Opt. Eng. **25**, 1075 (1986).
11. D. Psaltis and N. Farhat, "Optical Information Processing Based on an Associative-Memory Model of Neural Nets with Thresholding and Feedback," Opt. Lett. **10**, 98 (1985).
12. N. H. Farhat, D. Psaltis, A. Prata, and E. Paek, "Optical Implementation of the Hopfield Model," Appl. Opt. **24**, 1469 (1985).
13. J. L. Jewell, Y. H. Lee, M. Warren, H. M. Gibbs, N. Peyghambarian, A. C. Gossard, and W. Wiegmann, "3-pJ, 82 MHz Optical Logic Gates in a Room Temperature GaAs-AlGaAs Multiple-Quantum Well Etalon," Appl. Phys. Lett. **46**, 918 (1985).
14. M. F. Bone, D. Coates, W. A. Crossland, P. Gunn, and P. W. Ross, "Ferroelectric Liquid Crystal Display Capable of Video Line Address Times," Displays **1**, 115 (1987).

CHAPTER 6

"Multitarget Tracking with an Optical Neural Net Using a Quadratic Energy Function"

Proc. SPIE
Vol. 1192
November 1989

MULTITARGET TRACKING WITH AN OPTICAL NEURAL NET USING A QUADRATIC ENERGY FUNCTION

Mark Yee, Etienne Barnard and David Casasent

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh PA 15213

Abstract

Multitarget tracking over consecutive pairs of time frames is accomplished with a neural net. This involves position and velocity measurements of the targets and a quadratic neural energy function. Simulation data are presented, and an optical implementation is discussed.

Introduction

Since its original inception the Hopfield neural network [1] has been used to solve a variety of problems, including associative memories and multitarget tracking. A prior multitarget tracking neural net tracked each set of three consecutive time frames using a cubic energy function [2]. We now consider a similar realization utilizing the Hopfield net and a quadratic energy function that leads to a simpler optical implementation. These solutions are preferable to other multitarget tracking neural nets that require one neuron per image pixel [3].

Neural Energy Function

The problem is to track multiple targets over two consecutive time frames. The position and velocity vectors of each target are assumed known via real-time measurements. It is further assumed that there is a one-to-one correspondence between the number of measurements and the number of targets in each time frame. That is, a given position and velocity vector pair is due to no more than one target, and a given target will produce only one position and one velocity vector. Unresolvable crossing targets and spurious measurements can still be handled if these assumptions are not true, as our initial results indicate. They are only invoked for simplicity in this particular investigation. Finally, it is assumed that no acceleration occurs between the two time frames so that velocity is constant for each target over the time span between samples.

The interconnection pattern linking the measurements of one frame with those of another is unique due to the one-to-one correspondence condition. The problem is solved with a neural net by assigning each neuron X_{ij} to one of the interconnections between measurements i and j in the two frames. With N measurements per frame there are N^2 neurons required. The relative activity of each neuron indicates the validity of each connection between measurements in the two frames. Ideally, only those neurons associated with valid connections will be active, while all others are driven to zero by minimization of an appropriate neural energy function. One such energy function is given by

$$E(X) = C_1 \sum_i \sum_j X_{ij} D_{ij} + C_2 \sum_i (\sum_j X_{ij} - 1)^2 + C_3 \sum_j (\sum_i X_{ij} - 1)^2 \quad (1)$$

This is a quadratic neural energy equation which, when minimized, will determine the best measurement assignments. The coefficients were $C_1 = C_2 = C_3 = 1$ in this case, but in general can be varied to weight the terms differently. Each term will now be explained.

The bias term D_{ij} is derived from the position and velocity measurements in both time frames. Let $\vec{P}_{11}$ be the first measured position vector in the first time frame, and $\vec{P}_{12}$ be the first measured position vector in the second

time frame. Also let the corresponding measured velocity vectors be represented by $\vec{V}_{11}$ and $\vec{V}_{12}$ respectively. The D_{11} term is then

$$D_{11} = A\|\vec{P}_{11} - \vec{P}_{12}\|^2 + B\|\vec{V}_{11} - \vec{V}_{12}\|^2 \qquad (2)$$

While the multiplicative coefficients A and B may in general be different, they were made equal to give the position and velocity measurements equal weight. The D_{ij} term is a measure of the difference in the position and velocity of the ith measurement in the first time frame and the jth measurement in the second time frame. Likewise the X_{ij} neuron corresponds to the connection between the ith and jth measurements. The first term in (1) thus serves to weaken the neurons which correspond to the largest (magnitude) changes in position and velocity. The last two terms in (1) are equally weighted, therefore the relative weights of the first term and the last two can be adjusted by the choice of A and B in (1).

The last two terms in (1) enforce the condition of one-to-one correspondence between measurements in the two frames. For every measurement i in the first frame there is no more than one corresponding measurement in the second frame, and for every measurement j in the second frame there is no more than one corresponding measurement in the first frame. This still allows the absence of a corresponding measurement or more measurements in one frame than the other, in which case all neurons associated with the "extra" measurement will be driven to zero. This accomodates the scenario where a target just enters or leaves the field of view during one of the time frames. The final X solution should ideally have only N ones in it, where N is the minimum of the number of measurements in each of the two frames.

In the Hopfield neural net model, the time evolution of the neurons from discrete time step $t = n$ to $t = n+1$ can be given by

$$X_{kl}(n+1) = X_{kl}(n) - \eta \Delta X_{kl} \qquad (3)$$

where n is the discrete time index and η is the step size. From the derivative of (1), then

$$\Delta X_{kl} = \frac{\partial E(X)}{\partial X_{kl}} = D_{kl} + 2(\sum_j X_{kj} - 1) + 2(\sum_i X_{il} - 1) \qquad (4)$$

where subscripts k and l have been introduced for clarity. Each neuron value is updated by iteratively subtracting ΔX_{kl} from each X_{kl} until the net converges to a solution.

Matrix-Vector Formulation

The doubly-subscripted neurons X_{ij} can be thought of as elements of a matrix, with i as the row index and j as the column index. Each position in the matrix then represents a unique connection between measurements in the two time frames. The bias terms D_{ij} can be thought of in a similar manner. This representation is convenient for verifying convergence, as the criteria of one-to-one correspondence is equivalent to having only one maximum neuron value per row and per column. The indices of each maximum (i and j) yield the desired interconnection information between the time frames. However, actual implementation of the network is far easier if the X and D values are described as elements of one-dimensional matrices, or vectors, with elements $\tilde{X}_i$ and $\tilde{D}_i$. Specifically, the first N elements of the $\tilde{X}$-vector are the X_{1j} terms, the next N elements are the X_{2j} terms, etc. , assuming there are N measurements in each time frame; a similar column vector is used for the D_{ij} terms.

Using the vector format, we implement the neural network in (4) by a matrix-vector multiplication and a vector addition,

$$\Delta \tilde{X}_i = \eta(\sum_j M_{ij}\tilde{X}_j + \tilde{D}_i). \qquad (5)$$

The weight matrix M in (5) combines the two summations in (4), as we now detail by considering the X_{kl} terms in (4). For a given k, the first summation in (4) is the sum of the elements contained in row k of the X matrix. Likewise, for a given l, the second summation is the sum of all elements in column l of the matrix. Now consider the matrix-vector product of M and $\tilde{X}$. Both summations are satisfied for the case of 9 neurons by

$$M = \begin{bmatrix} 4 & 2 & 2 & 2 & 0 & 0 & 2 & 0 & 0 \\ 2 & 4 & 2 & 0 & 2 & 0 & 0 & 2 & 0 \\ 2 & 2 & 4 & 0 & 0 & 2 & 0 & 0 & 2 \\ 2 & 0 & 0 & 4 & 2 & 2 & 2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 4 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 2 & 2 & 4 & 0 & 0 & 2 \\ 2 & 0 & 0 & 2 & 0 & 0 & 4 & 2 & 2 \\ 0 & 2 & 0 & 0 & 2 & 0 & 2 & 4 & 2 \\ 0 & 0 & 2 & 0 & 0 & 2 & 2 & 2 & 4 \end{bmatrix} \qquad (6)$$

where the factor of 2 in each summation in (4) has been included in M. In general this same block structure can be extended to accomodate more targets, where there are $N = 3$ targets and N^2 neurons in the above example. For the case of an unequal number of measurements (N_1

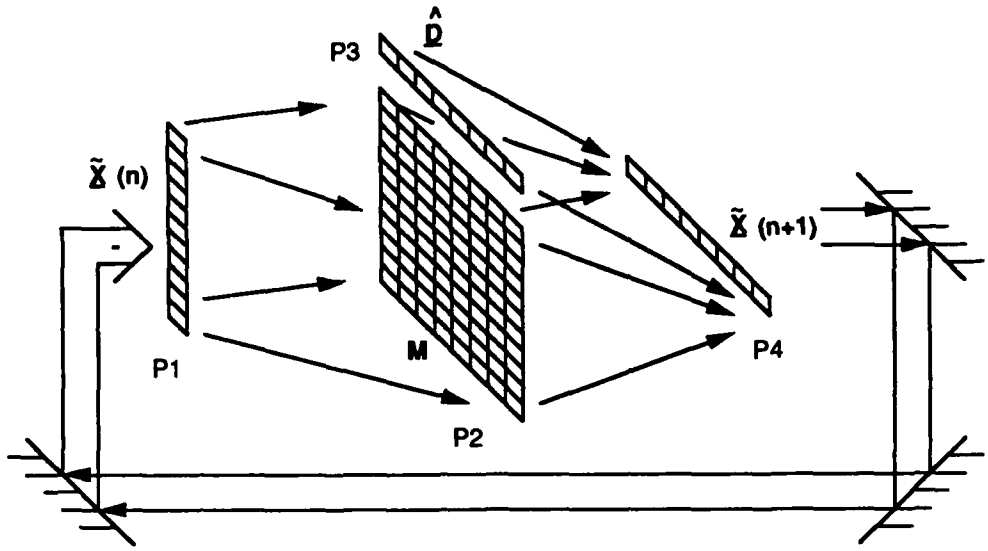


Figure 1: Optical implementation.

measurements in the first frame and N_2 measurements in the second), the same block matrix structure of M is retained, with each block matrix being $N_2 \times N_2$, and with N_1 block matrices in each row and column of M.

The -2 terms in each summation are combined with $\tilde{D}_i$ to give

$$\hat{D}_i = \tilde{D}_i - 4 \tag{7}$$

as in (5). Thus, (5) can be implemented as a matrix-vector multiplier with an added bias vector $\hat{D}_i$. This lends itself to an optical implementation as shown in Fig. 1. The neurons $\tilde{X}$ are represented by a linear array of laser diodes or a linear spatial light modulator in plane P1. The weight matrix M is the two-dimensional matrix in P2, which is fixed and can be stored on film. The bias vector $\hat{D}$ is produced by another array of laser diodes or 1-D spatial light modulator in P3. The summed result is detected at P4 and fed back electronically to the P1 array for the next iteration. A nonlinear function (often referred to as the neuron function) is applied to the P4 output in Fig. 1 to keep the $\tilde{X}_i$ values between zero and one. The nonlinear function we used is the sigmoid function

$$\tilde{X}_i = 0.5(1 + \tanh \frac{u_i}{u_0}), \tag{8}$$

where the u_i are the detected values at P4 and u_0 is a parameter which determines the slope of the function, or the degree of "binarization", of the neurons.

Simulation Results

One of the more difficult scenarios for multitarget tracking involves multiple crossing targets. A set of six crossing targets (denoted by A to F) with 3-D position and velocity vectors was created for use in simulation of the network. A total of five time frames (four pairs of measurments) were processed in the initial simulations. Fig. 2 shows the target positions in time frames 3 and 4. The target position vectors are shown projected onto the three (x-y, x-z, y-z) major planes. Each of the six targets are represented by an arrow with the tail denoting the position in frame 3 and the head denoting the position in frame 4. The direction of the arrows indicates the target velocity vector direction, and the length of the arrow indicates the velocity vector magnitude. Target A is moving in the x-y plane along the x-axis, and thus its projection in the y-z plane is a point at the origin. Similarly, targets B and C are traveling along the y and z axes respectively (and thus appear as points at the origin in the x-z and x-y planes respectively). All six targets cross at the origin at different times between time frames 3 and 4.

Fig. 3 shows the x-y projections of the targets at time frames 3, 4 and 5. Each target is denoted by a separate symbol with its position in time frame 3 denoted by the symbol labeled with a letter. For example, target A

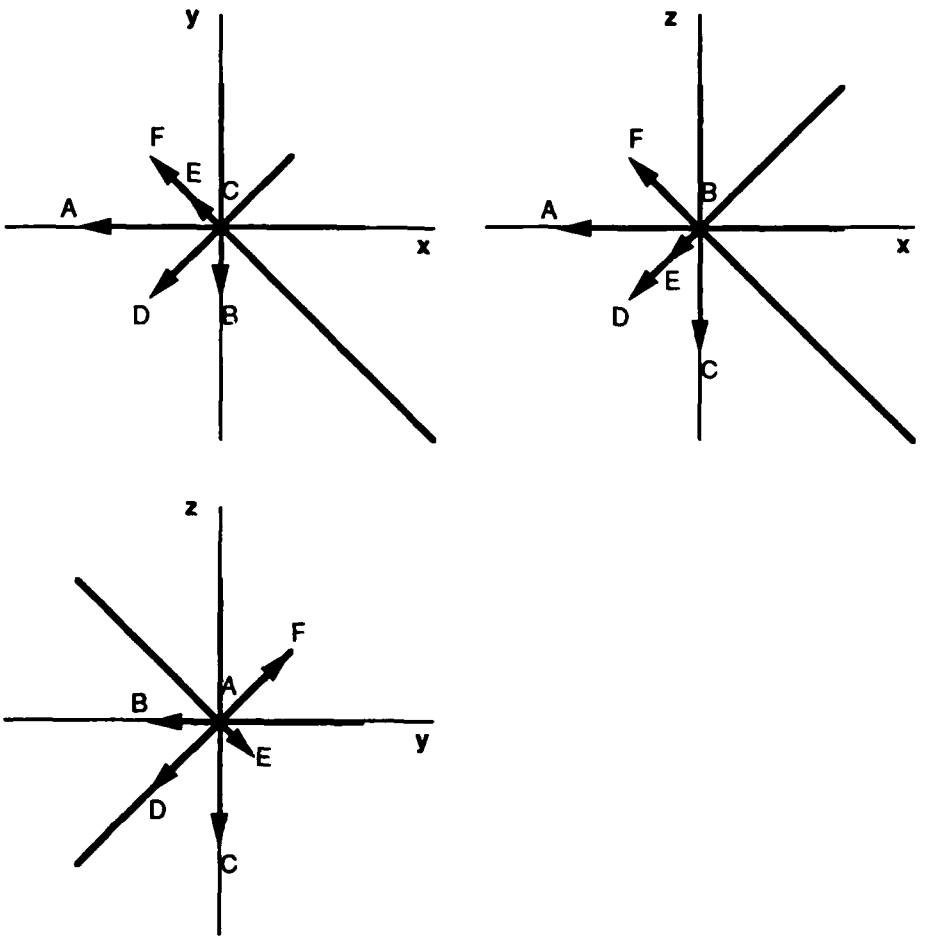


Figure 2: 2-D projections of 3-D position vectors for the targets in our scenario.

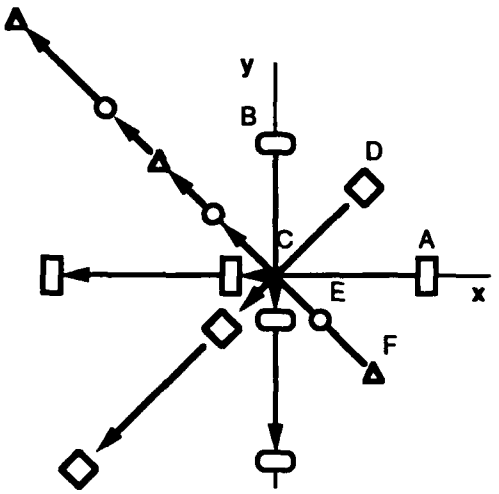


Figure 3: Time sequence of target positions (projected in the x-y plane) for three successive time frames.

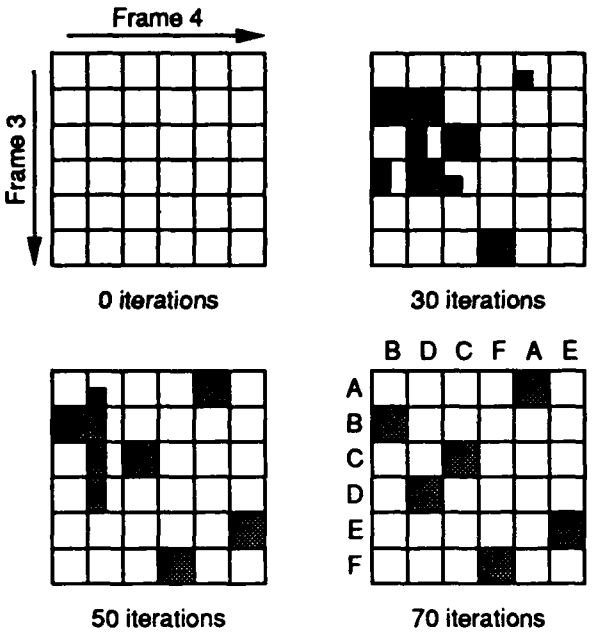


Figure 4: Convergence of neuron values.

moves right to left along the x-axis as shown. Target C travels in the negative z direction (into the page) and thus appears as a point at the origin in all three time frames. The arrows indicate velocity vector direction. The scale of the axes in Fig. 3 differs from Fig. 2, in order to allow inclusion of all targets over the three time frames.

The simulation of our algorithm was performed on a Hecht-Nielsen neurocomputer system using Anza Plus neurosoftware. This system allows direct implementation of the Hopfield neural network and other major neural net models. The bias vector $\hat{D}$ and the weight matrix M were computed offline and loaded into the neurocomputer. A new vector $\hat{D}$ is used for each pair of frames, while the weight matrix M remains the same for all pairs of frames. The network iterated until the neuron values converged (equal to within .01 for at least two consecutive iterations). The initial neuron states were randomized using a uniform zero-mean distribution with a maximum value of ± 0.0003. Prior to randomization the initial neuron states were all set equal to $1/36$ so that the sum of the $N^2 = 36$ neurons was equal to one. Ten different initial neuron vectors were used for each of the four pairs of time frames. (The iteration data were averaged over the ten runs, and negligible differences were obtained). A graphical representation of the neuron states X_{ij} in the two-dimensional matrix format is shown at four iteration steps in Fig. 4 for time frames 3 and 4. The indices in Fig. 4 are i = frame 3 (vertical) and j = frame 4 (horizontal) with the six measurements per frame assigned randomly. The size of the darkened squares indicates the neuron value, with a clear square denoting a value of zero, and a fully darkened square denoting a value of one. The initial (iteration 0) neuron values are too small to be resolved in the figure, since the maximum random variance is small. Successive iterations lead to the final stable pattern shown at iteration 70 for a single pair of time frames. This final X_{ij} pattern illustrates the interconnection assignments of corresponding measurements in the two frames. (The indices associated with targets A to F in the two frames are shown in this final matrix). A single one is present in each row and column, and they correctly associate the targets in the two frames. The smallest "true" neuron activity level was 0.924 and the largest "false" level was 0.01 (thus allowing easy thresholding).

Simulations were run for two different sequences of six targets. Case 2 was the scenario illustrated in Figs. 2-3. Case 1 is a less complicated crossing scenario (with only two or three targets crossing at any one time). To simulate realistic cases, random noise was added to the measured position and velocity vectors in (2) for each target in frames 2 to 5. This measurement noise was uniformly distributed with zero mean and a maximum percent error in any measured position or velocity component of 0%, 5% and 10%. The average number of it-

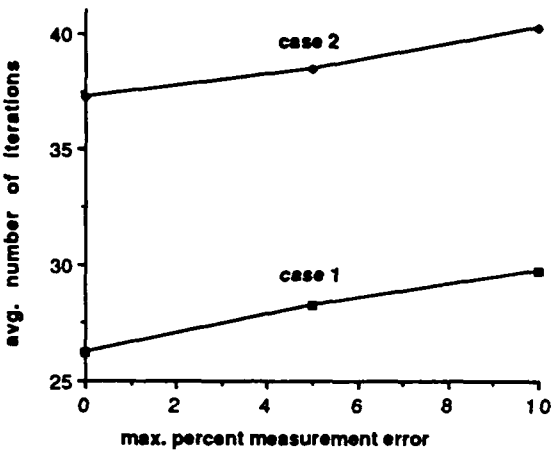


Figure 5: Iterations vs. measurement error.

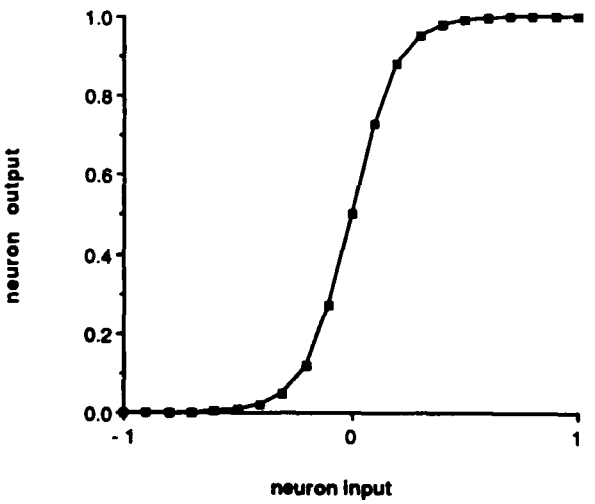


Figure 6: Neuron function for $u_0 = 0.2$.

erations required (over the four pairs of time frames and ten initial neuron vectors) for the different amounts of measurement noise in the two cases are shown in Fig. 5. As expected, the simpler case 1 scenario requires fewer iterations. The number of iterations required for any pair of frames ranged from 15 to 40 for case 1 and from 20 to 70 for case 2 (with the largest number of iterations, 70 as in Fig. 4., corresponding to the time frame when all six targets crossed with the maximum 10% measurement error present). The results show that even in the presence of measurement errors, the neural net was very robust and did not require a significant increase in the average number of iterations (i. e. 26 to 30 iterations for case 1 and 37 to 40 iterations for case 2). In all cases, the network converged to the correct solution for all six targets in each of the time frames.

The network parameters used were $\eta = 0.3$ and $u_0 = 0.2$ in all runs. A graph of the neuron function for $u_0 = 0.2$ is shown in Fig. 6. No intensive effort was made to optimize these parameters. This merits future work. For example, the neuron increment in (4) is linear in X_{ij}, and for this case techniques for choosing η exist [4]. Extending these to piecewise-linear cases may be possible. Smaller u_0 values increase the "binarization" of the neuron values and increase the probability of convergence to relatively shallow local minima rather than the correct global minimum. Larger u_0 values make thresholding of the outputs more difficult. The $u_0 = 0.2$ value used is a compromise between these effects.

The A and B parameters in (2) used to determine the bias vector were equal and constant for each case. We used $A = B = 0.01$ for all time frames in case 1, and $A = B = 0.025$ for case 2. Larger values for these parameters weight the bias term (due to the position and velocity vectors) more than the one-to-one correspondence terms in the energy function. For case 2, the larger A and B values were used to improve the rate at which the network converged. (Larger values also increased the probability of an incorrect solution). The values used are not optimized in these initial tests.

Conclusion

A Hopfield neural net using a quadratic cost function was successfully used to solve the multitarget tracking problem over pairs of consecutive time frames using 3-D target trajectories. This network can be implemented with a simple optical architecture to achieve real-time processing for large numbers of targets, with convergence in rel-

atively few iterations (15-70). With optimization, faster convergence times are expected. The network proved to be robust, converging successfully with measurement errors of up to 10 percent.

Acknowledgments

We gratefully acknowledge the support of both the Strategic Defense Initiative Office of Innovative Science and Technology monitored by the Office of Naval Research, and the Defense Advanced Research Projects Agency monitored by the U. S. Army Missile Command. All simulations were performed on a Hecht-Nielsen digital neurocomputer.

References

[1] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model," *Science*, vol. 233, pp. 625–633, 1986.

[2] E. Barnard and D. Casasent, "Multitarget tracking with cubic energy optical neural nets," Accepted for publication in Applied Optics, 1989.

[3] R. M. Kuczewski, "Neural network approaches to multi-target tracking," in *Proc. IEEE ICNN*, June 1987, pp. 619–633.

[4] B. Widrow, J. M. McCool, M. G. Larimore, and C. R. Johnson, "Stationary and nonstationary learning characteristics of the LMS adaptive filter," *Proc. IEEE*, vol. 64, pp. 1151–1162, August 1976.

CHAPTER 7

"A Symbolic Neural Net Production System: Obstacle Avoidance, Navigation, Shift-Invariance and Multiple Objects"

Proc. SPIE
Vol. 1195
November 1989

A SYMBOLIC NEURAL NET PRODUCTION SYSTEM: OBSTACLE AVOIDANCE, NAVIGATION, SHIFT-INVARIANCE AND MULTIPLE OBJECTS

David Casasent
and Elizabeth Botha

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

ABSTRACT

A symbolic neural net is described. It uses a multichannel symbolic correlator to produce input neuron data to an optical neural net production system. It has use in obstacle avoidance, navigation, and scene analysis applications. The shift-invariance and ability to handle multiple objects are novel aspects of this symbolic neural net. Initial simulated data are provided and symbolic optical filter banks are discussed. The neural net production system is described. A parallel and iterative set of rules and results for our case study are presented. Its adaptive learning aspects are noted.

1. INTRODUCTION

Figure 1 shows an overview of the symbolic neural net. A multichannel correlator (Section 2) analyzes the input scene. It is unique since it can handle multiple objects. It provides a symbolic description of the input scene. This is converted to a position encoded input neuron description that indicates what basic elements are present in each region of the scene (Section 3). This is a new neuron representation space. No prior neural net can accommodate multiple objects in the field of view (FOV). A neural net production system (Section 4) then determines the contents of each spatial region of the scene. Sections 5 and 6 present initial simulation data using this system.

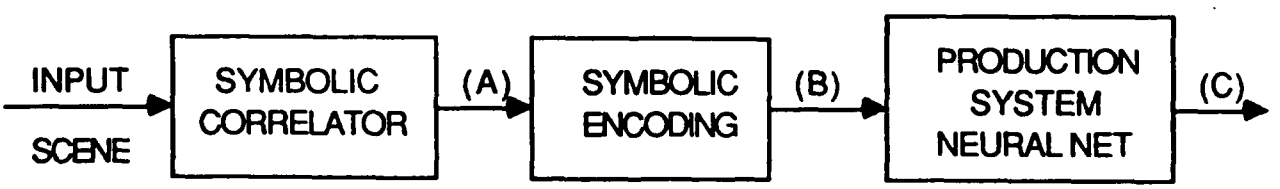


FIGURE 1. Optical symbolic neural net.

2. SYMBOLIC CORRELATOR

Figure 2 shows a 4-channel optical symbolic correlator [1]. P1 contains the input scene, 4 frequency-multiplexed filters are placed at P2 and 4 spatially separated correlation planes appear at P3. Each correlation plane contains peaks at the locations in P1 of the 4 different filter patterns used at P2 (thus shift invariance exists and multiple objects in the FOV can be handled). Only correlators provide this capacity in parallel. The 4 correlation plane patterns are read off point-by-point in parallel from top left to bottom right. A segmented CCD can achieve this. The output is a 4-digit symbolic word that describes the P2 filters' response for each spatial region of the scene at P1. With 4 binary encoded P2 filters (correlation peak values of 1 and 0), we can describe $2^4 = 16$ objects. With F filters and L correlation peak levels, we can encode $L^F = 10^4 = 10{,}000$ classes (this is an enormous potential). Multi-level encoding and distortion-invariant filters are possible and have been previously advanced [2]. Our production system neural net (Section 4) analyzes these symbolic outputs (for each spatial region) and thus we achieve the first shift-invariant multiple object neural net. Shift-invariance is achieved by use of a correlator. High capacity is achieved by symbolic encoding. Distortion-invariance is achieved by the use of synthetic discriminant function (SDF) [2] filters. Extensions to F > 4 filters (digits) are possible.

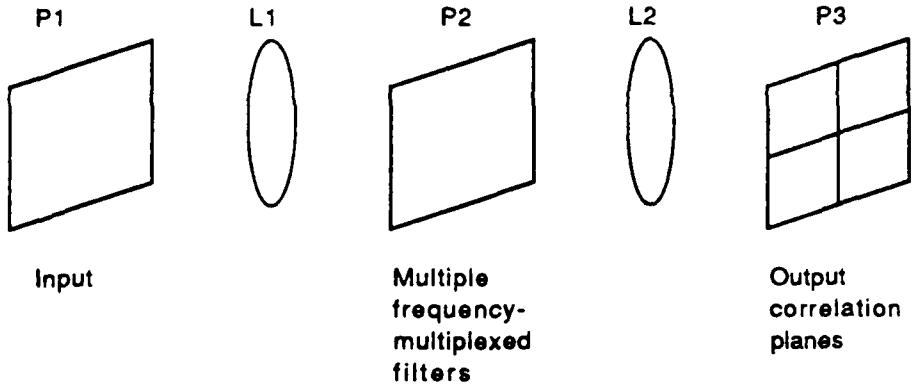


FIGURE 2. Frequency-multiplexed optical symbolic correlator for neural net representation generation.

3. NEURON REPRESENTATION

We show the four time sequential raster outputs from the 4 correlation planes (Figure 2) by F1 to F4 in Figure 3. These are our 4-digit descriptions of each P1 region. Our symbolic encoding system converts this into our position-encoded neuron description for input to our production system neural net as shown conceptually in Figure 4. In the specific input neuron representation we consider, each input neuron to our production system is position encoded to represent a generic object part.

4. NEURAL NET PRODUCTION SYSTEM CONCEPT

A production system consists of IF-THEN statements. Its realization is possible via a neural net or a symbolic substitution system [3]. Here, we consider its neural net realization. For our present problem, such a rule-based system is necessary to determine the input present at each spatial region of P1 from the 4-digit symbolic data of Figure 3 position-encoded as in Figure 4. We describe this concept by example. Figure 5 shows a simple set of 4 rules with antecedents on the left and

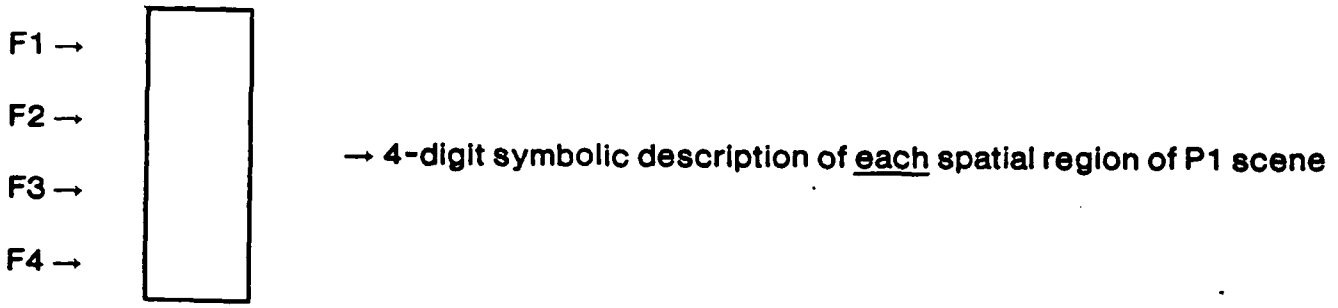


FIGURE 3. Symbolic F1 to F4 encoding from Figure 2 correlator.

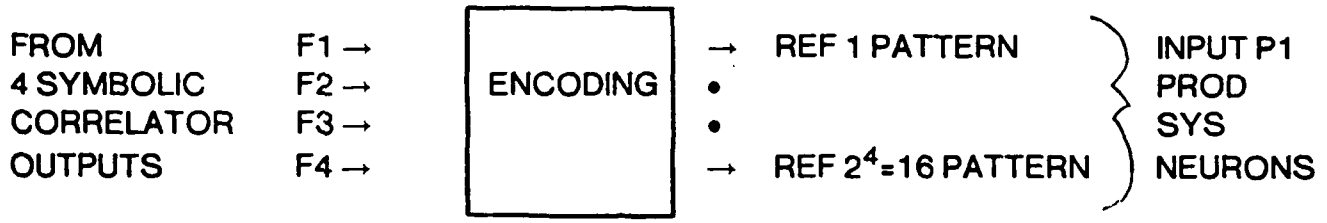


FIGURE 4. Neuron representation (position encoding) of symbolic optical correlator F1 to F4 outputs.

consequents on the right. We write all rules as IF-THEN statements. We allow the AND of various antecedents and we allow (Figure 6) the OR of several such sets of antecedents. The antecedents (a_n) are facts known to be true. The output consequents (c_n) are new true facts. If we denote each fact (antecedent or consequent) by a neuron in a specific position (location), then the rules can be described as weighted combinations of input neurons (true facts are input or output neurons that are active "1" and false facts are neurons with activation "0"). Figure 7 shows the neural net that realizes the rules in Figure 5. Figure 8 shows a standard optical matrix-vector multiplier that realizes the production system neural net of Figure 7. The input facts (neurons) are represented by activated point modulators at P1 (LEDs, laser diodes, etc.). The weights (rules) are the elements of an interconnection matrix at P2 and the output (antecendents) facts are activated detector elements at P3. The P2 weights or P3 thresholds are adjusted to produce proper outputs [3]. The diagonal elements at P2 are one so that input facts remain true. New rules not present in the original rule base can be inferred and operation on facts with various degrees of confidence are possible as we have detailed [3].

5. SIMULATION RESULTS

For obstacle avoidance applications, we require only the relative size of the objects (bolders etc.) in each spatial region of P1. For navigation, we must identify what exists in each P1 region and relate it to a global map of the scene. For general scene analysis, we desire to know what is present (object name) in each region of P1.

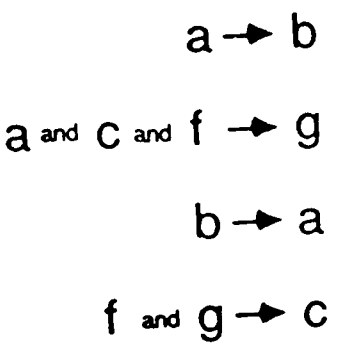


FIGURE 5. Simple if-then production rules.

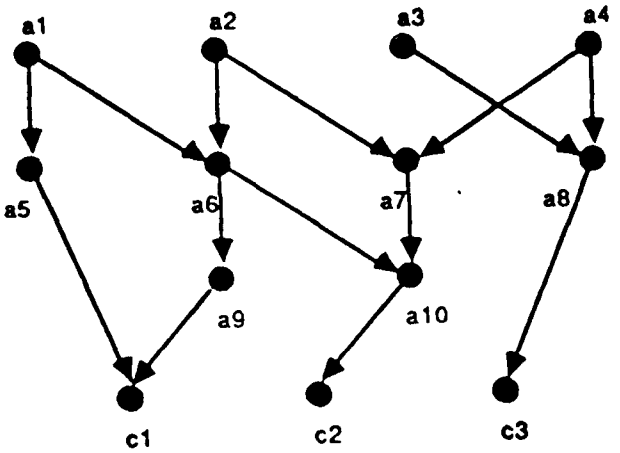


FIGURE 6. Production system with the OR of several paths to the same consequent c_n.

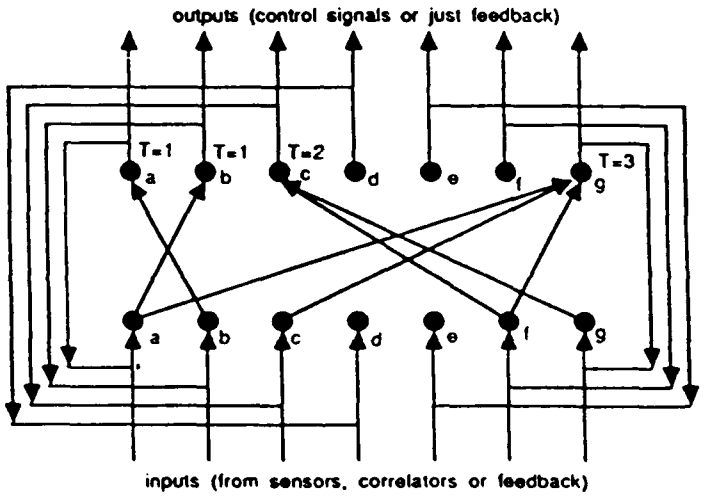


FIGURE 7. Neural net for the rules in Figure 5.

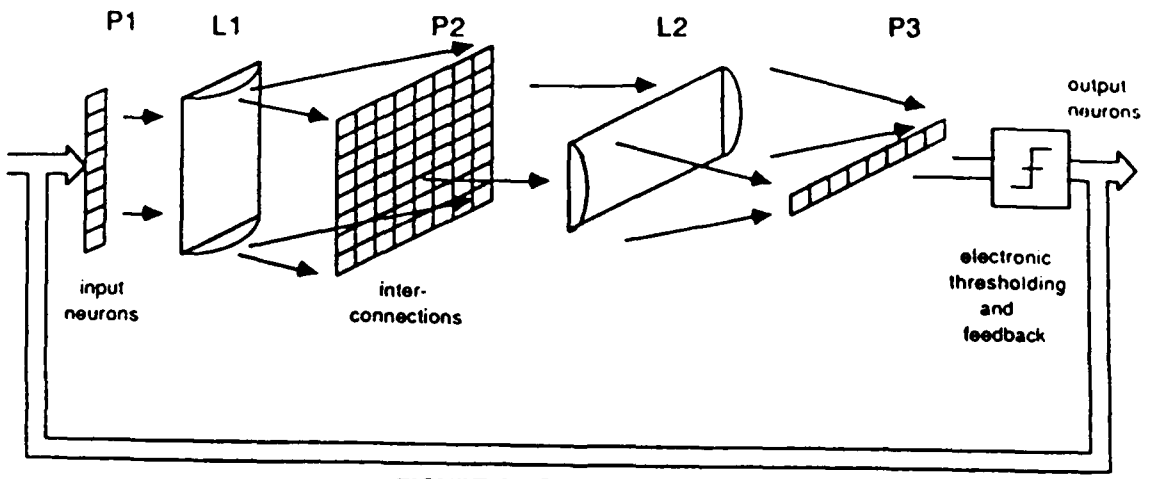


FIGURE 8. Optical neural net.

5.1 Database

To demonstrate all aspects of such a neural net, we consider a database consisting of 9 objects (Figure 9) with 2 objects shown enlarged for clarity. We used a synthavision solid model description in which each object consists of basic general shapes or parts. Figure 10 shows the 12 object parts we consider. Thus, the version of the general problem that we address is the location and recognition of each object part in the input scene and the subsequent identification of what object is present from this object part information. We use an optical correlator (Figure 2) to locate the object parts present and we use an optical neural net production system (Figure 8) to determine what object is present. Table 1 lists the 9 objects we consider. Table 2 lists the parts we used to synthesize each. Table 3 lists the 3 clusters into which we grouped these objects. Table 4 lists the parts we use within each cluster to uniquely describe the objects within that cluster.

We formed SDF filters for each object part. To test the invariance of these filters, we formed 3 and 4 distorted versions of each object and of each object part. These images corresponded to 0°, 30°, 60° and 90° (or variations thereof as noted) rotated aspect versions of the original (Figures 9 and 10) objects and parts. We formed projection SDFs [2] from the 4 different aspect views of each object part and tested each such filter against the test set of distorted objects.

5.2 Distortion-Invariant Part Recognition

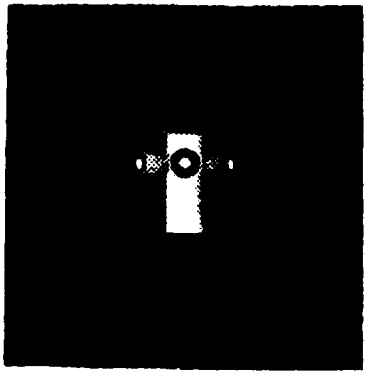
The results in Tables 5-7 show that the object parts within each cluster can adequately discriminate the object parts used and that they can be recognized invariant to distortions. The underlined entries in these tables of data denote the object parts denoted as being present in the object indicated under test (with proper thresholds used). These results show that the correlation filters (one per part) can identify all tested object parts independent of position and distortion. The "tank" object was a poor choice and was not used.

6. NEURAL NET PRODUCTION SYSTEM

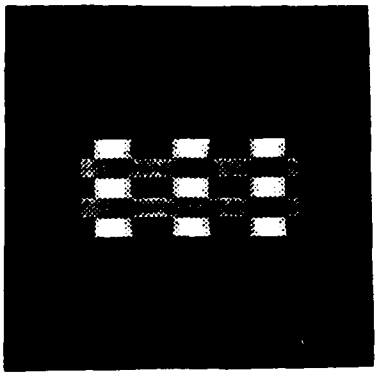
We used the results of Section 5 for our parts list and our object tests, and our production system concept (Section 4) to devise a parallel (Figure 11) and an iterative (Figure 12) neural net production system design. We tested these neural net production systems on our database with successful results.

Summary

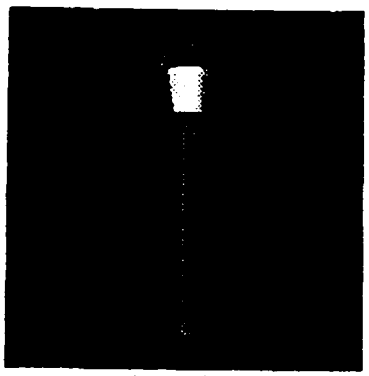
In this paper we have unified our prior symbolic correlator [1] and production system [3] neural net work into a symbolic neural net. It provides the ability to handle multiple objects in the field of view. It is shift-invariant, distortion-invariant and has the potential for high capacity. Initial simulation results were presented.



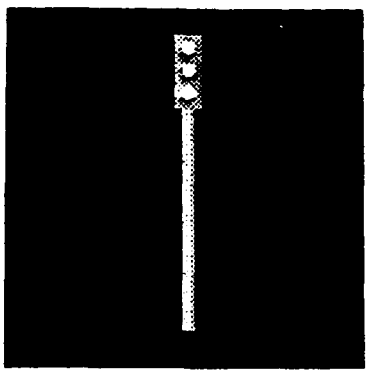
c_{fire} = fire hydrant



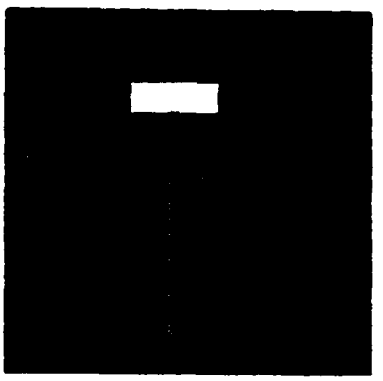
c_{fence} = fence



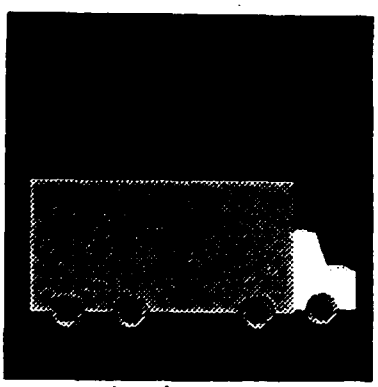
c_{lamp} = street lamp



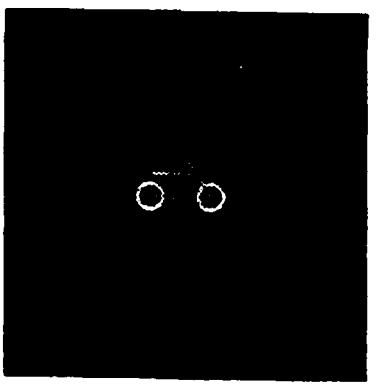
c_{tlight} = traffic light



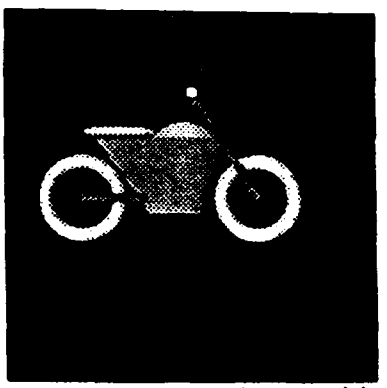
c_{sign} = street sign



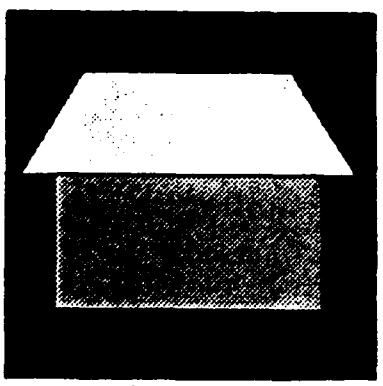
c_{truck} = truck



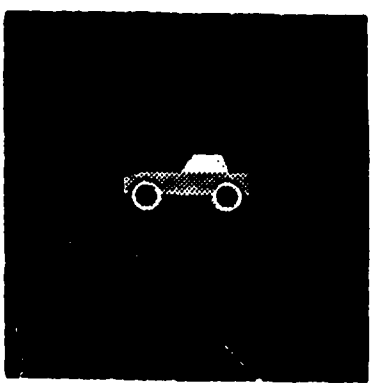
c_{motorc} = motor cycle



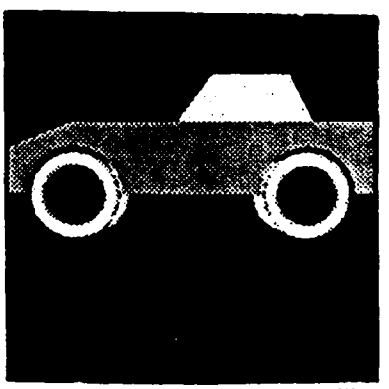
big motor cycle (to provide detail)



c_{house} = house

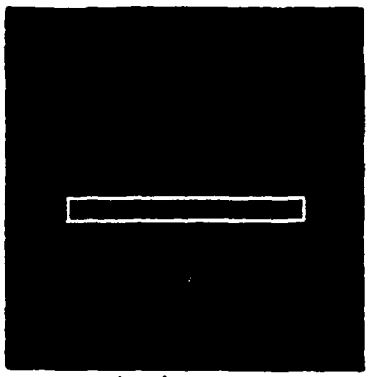


c_{car} = car

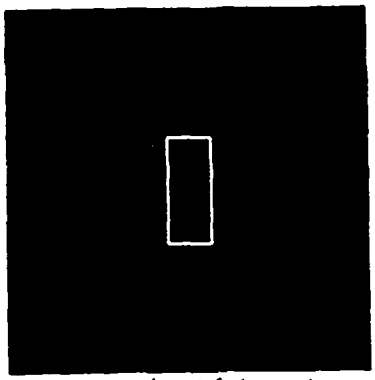


big car (to provide detail)

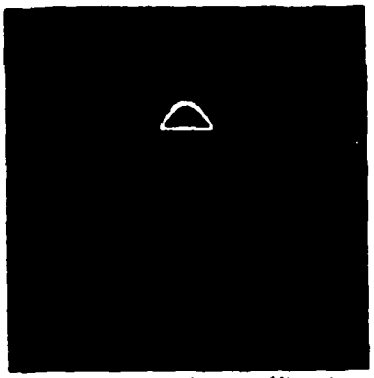
FIGURE 9. The 9 objects (and scaled versions of 2) in our initial production system.



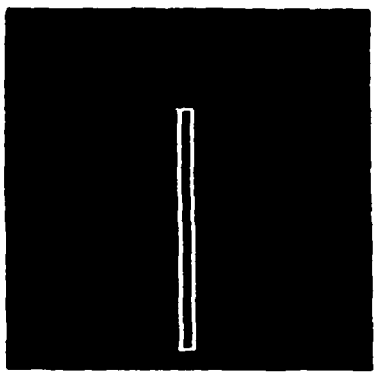
a_{hbar} = horizontal bar



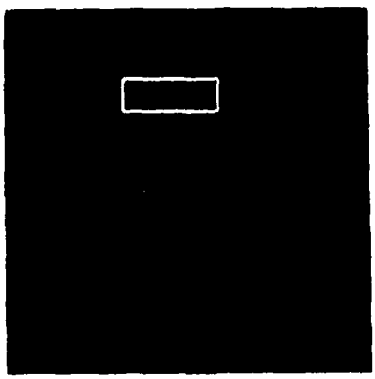
a_{sfpost} = short fat post



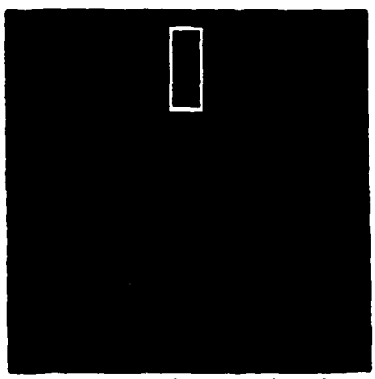
a_{dome} = top dome (fire hydrant)



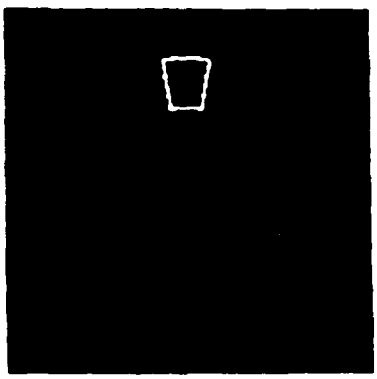
a_{post} = long thin post



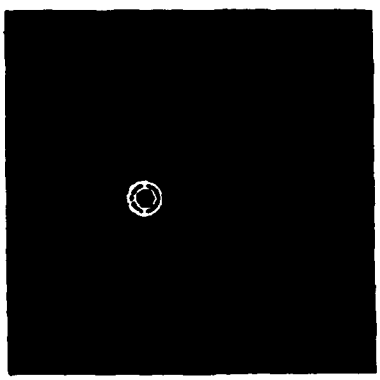
a_{name} = name plate (street sign)



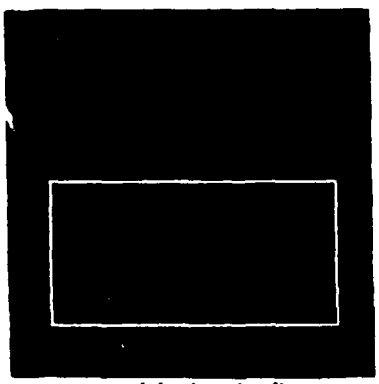
a_{tbox} = box (traffic light)



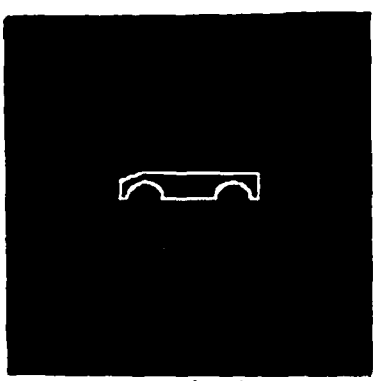
a_{light} = light (street lamp)



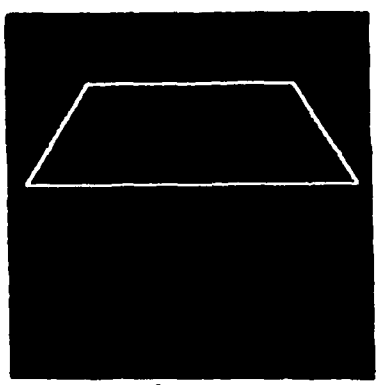
a_{wheel} = wheel



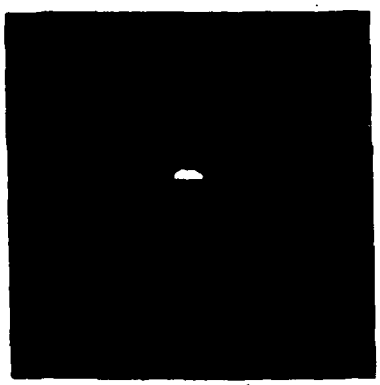
a_{bigbod} = big body (house, truck)



a_{carbod} = car body



a_{roof} = roof



a_{tank} = gas tank (motor cycle)

FIGURE 10. *The 12 parts used to describe as facts the multiple objects (Figure 9) in our database.*

c_{fire} = fire hydrant	c_{fence} = fence	c_{lamp} = street lamp
c_{tlight} = traffic light	c_{sign} = street sign	c_{car} = car
c_{truck} = truck	c_{house} = house	c_{motorc} = motor cycle

TABLE 1: Database of 9 Objects

Fire hydrant:	short fat post, dome on top, 3 arms
Fence:	3 short fat posts, 2 horizontal bars
Traffic light:	long thin post, box, 3 lights
Street lamp:	long thin post, light, dome on top
Street sign:	long thin post, rectangular name plate
Motor cycle:	2 wheels, engine, gas tank, seat, pipes
Car:	car body, 4 wheels, wedge shaped car top
Truck:	big square body, 8 wheels, cabin
House:	big square body, wedge shaped roof

TABLE 2: Components Used to Construct Objects

CLUSTER 1:	SHORT OBJECTS (fire hydrant, fence)
CLUSTER 2:	TALL OBJECTS (traffic light, street sign and lamp)
CLUSTER 3:	BIG OBJECTS (motor cycle, car, house and truck)

TABLE 3: Multiple Object Clusters Used for First Separation of Objects

CLUSTER-1 PARTS:	short fat post, dome, horizontal bar
CLUSTER-2 PARTS:	long thin post, box, light, rectangular name plate
CLUSTER-3 PARTS:	gas tank, car body, wheel, big body, wedge shaped roof

TABLE 4: Symbolic Parts for each Object Cluster

Acknowledgment

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency monitored by the U.S. Army Missile Command.

	fence 0	fence 30	fence 60	fence 90	fire 0	fire 30	fire 60	fire 90
hbar	1.195	1.184	1.130	0.808	0.823	0.687	0.620	0.738
dome	0.634	0.678	0.661	0.492	0.885	0.885	0.885	0.885
sfpost	0.742	0.742	0.737	0.871	0.892	0.868	0.858	0.946

TABLE 5: Cluster 1 Cross Correlation Data

	lamp	tlight 0	tlight 20	tlight 40	tlight 60	sign 0	sign 20	sign 40	sign 60
light	1.000	0.386	0.514	0.475	0.402	0.440	0.423	0.394	0.386
tbox	0.692	1.063	1.049	1.046	0.903	0.682	0.679	0.686	0.682
name	0.747	0.624	0.704	0.670	0.613	0.995	1.008	1.069	1.106
post	0.683	1.000	0.989	0.989	0.989	0.667	0.667	0.667	0.667

TABLE 6: Cluster 2 Cross Correlation Data

	car 0	car 30	car 60	house 0	house 30	house 60
bigbod	0.355	0.374	0.312	1.177	0.981	1.063
carbod	1.097	1.085	1.020	0.622	0.578	0.625
roof	0.306	0.305	0.289	1.032	1.002	1.034
tank	0.981	1.000	0.944	0.611	0.667	0.722
wheel	1.829	1.868	1.485	0.780	0.778	0.793
	motorc 0	**motorc 30**	**motorc 60**	**truck 0**	**truck 30**	**truck 60**
bigbod	0.232	0.222	0.179	1.047	1.070	0.872
carbod	0.927	0.720	0.655	0.750	0.731	0.695
roof	0.240	0.241	0.212	0.564	0.521	0.462
tank	1.000	1.000	1.000	1.000	1.000	1.000
wheel	1.962	1.995	1.943	1.827	1.810	1.586

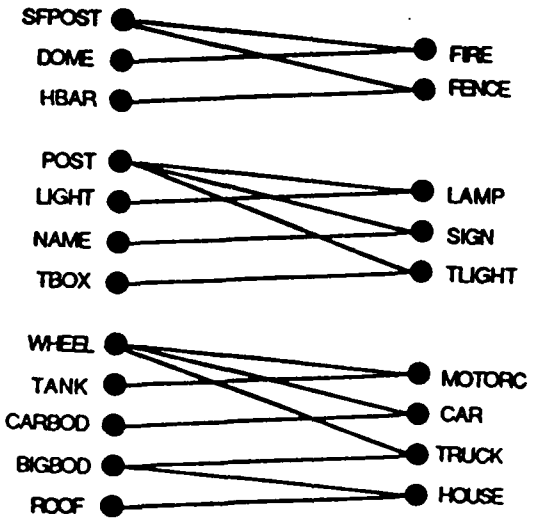
TABLE 7: Cluster 3 Cross Correlation Data

IF a_{sfpost} AND a_{dome} THEN c_{fire}
IF a_{sfpost} AND a_{hbar} THEN c_{fence}

IF a_{post} AND a_{light} THEN c_{lamp}
IF a_{post} AND a_{name} THEN c_{sign}
IF a_{post} AND a_{tbox} THEN c_{tlight}

IF $a_{wheel(s)}$ AND a_{tank} THEN c_{motorc}
IF $a_{wheel(s)}$ AND a_{carbod} THEN c_{car}
IF $a_{wheel(s)}$ AND a_{bigbod} THEN c_{truck}
IF a_{roof} AND a_{bigbod} THEN c_{house}

(a)



(b)

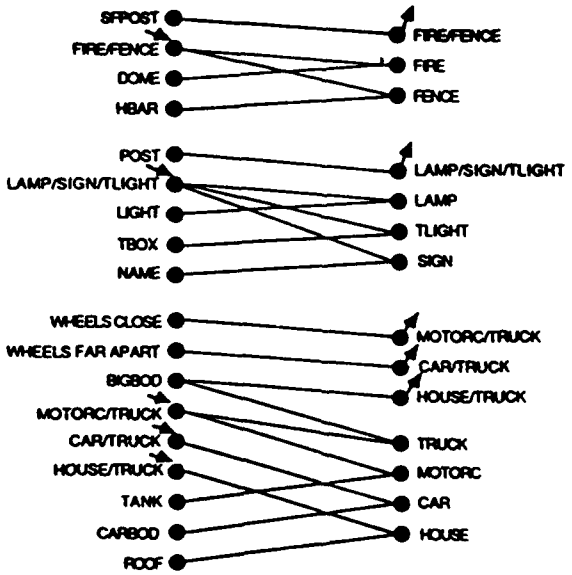
Figure 11. Rule base (a) and neural net (b) for a parallel production system.

IF a_{sfpost} THEN $c_{fire\ or\ fence}$
IF $a_{fire\ or\ fence}$ AND a_{dome} THEN c_{fire}
IF $a_{fire\ or\ fence}$ AND a_{hbar} THEN c_{fence}

IF a_{post} THEN $c_{lamp,\ sign\ or\ tlight}$
IF $a_{lamp,\ sign\ or\ tlight}$ AND a_{light} THEN c_{lamp}
IF $a_{lamp,\ sign\ or\ tlight}$ AND a_{tbox} THEN c_{tlight}
IF $a_{lamp,\ sign\ or\ tlight}$ AND a_{name} THEN c_{sign}

IF $a_{wheels\ close}$ THEN $c_{motorc\ or\ truck}$
IF $a_{wheels\ far\ apart}$ THEN $c_{car\ or\ truck}$
IF a_{bigbod} THEN $c_{house\ or\ truck}$
IF $a_{motorc\ or\ truck}$ AND a_{bigbod} THEN c_{truck}
IF $a_{motorc\ or\ truck}$ AND a_{tank} THEN c_{motorc}
IF $a_{car\ or\ truck}$ AND a_{carbod} THEN c_{car}
IF $a_{house\ or\ truck}$ AND a_{roof} THEN c_{house}

(a)



(b)

Figure 12. Rule base (a) and neural net (b) for an iterative production system.

References

1. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", Proc. SPIE, Vol. 625, pp. 220-225, January 1986.

2. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation", Applied Optics, Vol. 23, pp. 1620-1627, May 1984.

3. E. Botha, D. Casasent and E. Barnard, "Optical Production Systems Using Neural Networks and Symbolic Substitution", Applied Optics, Vol. 27, pp. 5185-5193, 15 December 1988.

CHAPTER 8

"Optical Laboratory Realization of a Symbolic Production System"

Proc. SPIE
Vol. 1295
April 1990

OPTICAL LABORATORY REALIZATION OF A SYMBOLIC PRODUCTION SYSTEM

David Casasent, Elizabeth Botha*, Jin Yun Wang and Ren Chao Ye

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

*Present Address:
Department of Electronics and Computer Engineering, University of Pretoria, South Africa

ABSTRACT

An optical symbolic neural net is described. It uses an optical symbolic correlator. This produces a new input neuron representation space that is shift-invariant and can accommodate multiple objects. No other neural net can handle multiple objects within the field of view. Initial optical laboratory data are presented. An optical neural net production system processes this new neuron data. This aspect of the system is briefly described.

1. INTRODUCTION

Figure 1 shows an overview of the symbolic neural net. A multichannel correlator (Section 2) analyzes the input scene. It is unique since it can handle multiple objects. It provides a symbolic description of the input scene. This is converted to a position encoded input neuron description that indicates what basic elements are present in each region of the scene (Section 3). This is a new neuron representation space. No prior neural net can accommodate multiple objects in the field of view (FOV). A neural net production system (Section 4) then determines the contents of each spatial region of the scene. Sections 5 and 6 present initial optical data using this system.

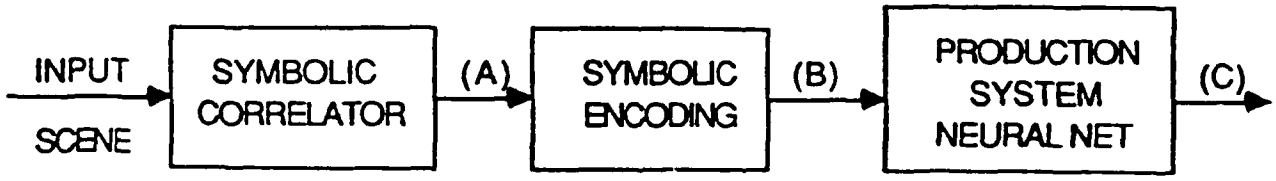


FIGURE 1. Optical symbolic neural net.

2. SYMBOLIC CORRELATOR

Figure 2 shows a 4-channel optical symbolic correlator [1]. P1 contains the input scene, 4 frequency-multiplexed filters are placed at P2 and 4 spatially separated correlation planes appear at P3. Each correlation plane contains peaks at the locations in P1 of the 4 different filter patterns used

at P2 (thus shift invariance exists and multiple objects in the FOV can be handled). Only correlators provide this capacity in parallel. The 4 correlation plane patterns are read off point-by-point in parallel from top left to bottom right. A segmented CCD can achieve this. The output is a 4-digit symbolic word that describes the P2 filters' response for each spatial region of the scene at P1. With 4 binary encoded P2 filters (correlation peak values of 1 and 0), we can describe $2^4 = 16$ objects. With F filters and L correlation peak levels, we can encode $L^F = 10^4 = 10{,}000$ classes (this is an enormous potential). Multi-level encoding and distortion-invariant filters are possible and have been previously advanced [2]. Our production system neural net (Section 4) analyzes these symbolic outputs (for each spatial region) and thus we achieve the first shift-invariant multiple object neural net. Shift-invariance is achieved by use of a correlator. High capacity is achieved by symbolic encoding. Distortion-invariance is achieved by the use of synthetic discriminant function (SDF) [2] filters. Extensions to F > 4 filters (digits) are possible. Section 4 provides initial optical laboratory results obtained with single and multiplexed filters.

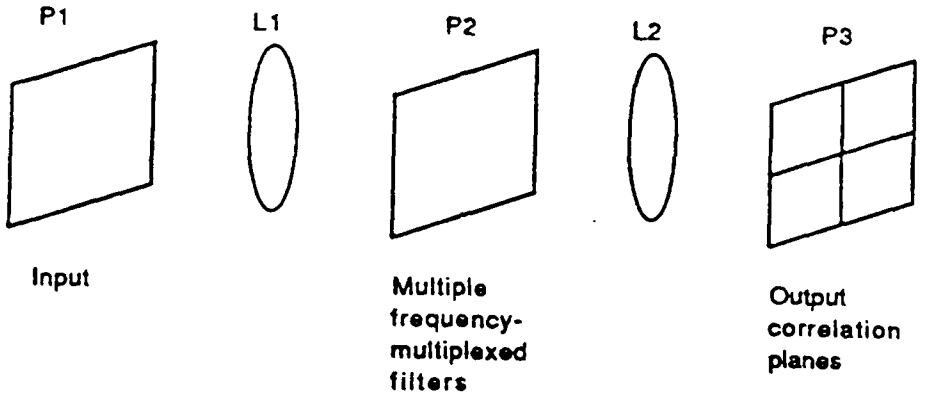


FIGURE 2. Frequency-multiplexed optical symbolic correlator for neural net representation generation.

3. NEURON REPRESENTATION

We denote the time sequential raster outputs from the 4 correlation planes (Figure 2) by F1 to F4 in Figure 3. These are our 4-digit descriptions of each P1 region. They are obtained in parallel for each P_1 region. Our symbolic encoding system converts this into our position-encoded neuron description for input to our production system neural net as shown conceptually in Figure 4. In the specific input neuron representation we consider, each input neuron to our production system is position encoded to represent a generic object part.

4. NEURAL NET PRODUCTION SYSTEM CONCEPT

A production system consists of IF-THEN statements. Its realization is possible via a neural net or a symbolic substitution system [3]. Here, we consider its neural net realization. For our present problem, such a rule-based system is necessary to determine the input present at each spatial region of P1 from the 4-digit symbolic data of Figure 3 position-encoded as in Figure 4. We describe this concept by example. Figure 5 shows a simple set of 4 rules with antecedents on the left and consequents on the right. We write all rules as IF-THEN statements. We allow the AND of various

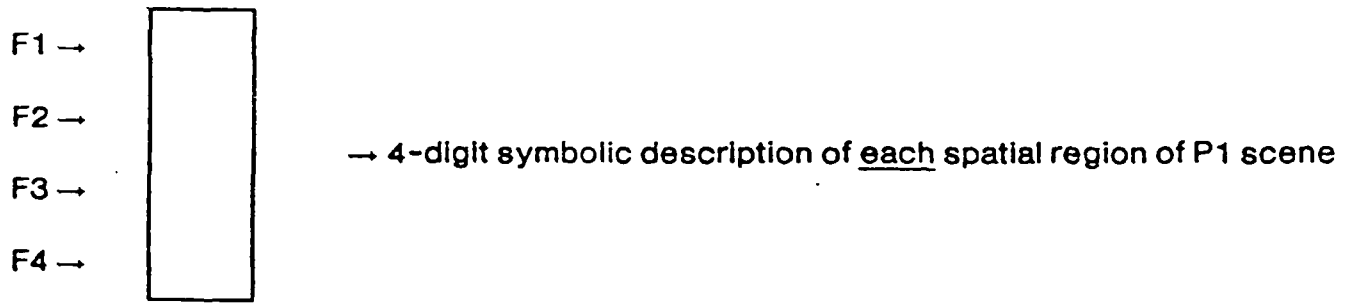


FIGURE 3. Symbolic F1 to F4 encoding from Figure 2 correlator.

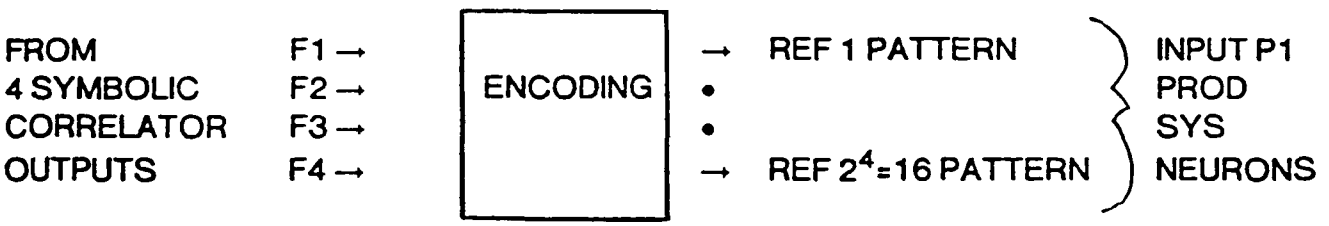


FIGURE 4. Neuron representation (position encoding) of symbolic optical correlator F1 to F4 outputs.

antecedents and we allow (Figure 6) the OR of several such sets of antecedents. The antecedents (a_n) are facts known to be true. The output consequents (c_n) are new true facts. If we denote each fact (antecedent or consequent) by a neuron in a specific position (location), then the rules can be described as weighted combinations of input neurons (true facts are input or output neurons that are active "1" and false facts are neurons with activation "0"). Figure 7 shows the neural net that realizes the rules in Figure 5. Figure 8 shows a standard optical matrix-vector multiplier that realizes the production system neural net of Figure 7. The input facts (neurons) are represented by activated point modulators at P1 (LEDs, laser diodes, etc.). The weights (rules) are the elements of an interconnection matrix at P2 and the output (antecendents) facts are activated detector elements at P3. The P2 weights or P3 thresholds are adjusted to produce proper outputs [3]. The diagonal elements at P2 are one so that input facts remain true. New rules not present in the original rule base can be inferred and operation on facts with various degrees of confidence are possible as we have detailed [3].

5. OPTICAL LABORATORY RESULTS

For obstacle avoidance applications, we require only the relative size of the objects (bolders etc.) in each spatial region of P1. For navigation, we must identify what exists in each P1 region and relate it to a global map of the scene. For general scene analysis, we desire to know what is present (object name) in each region of P1.

5.1 Database

To demonstrate this concept, we considered a database of 9 objects (Table 1). Each was

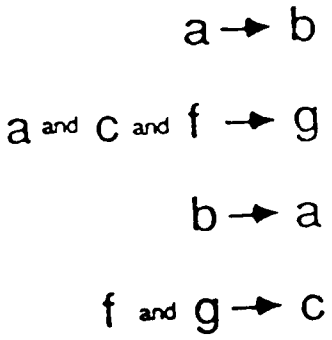


FIGURE 5. Simple if-then production rules.

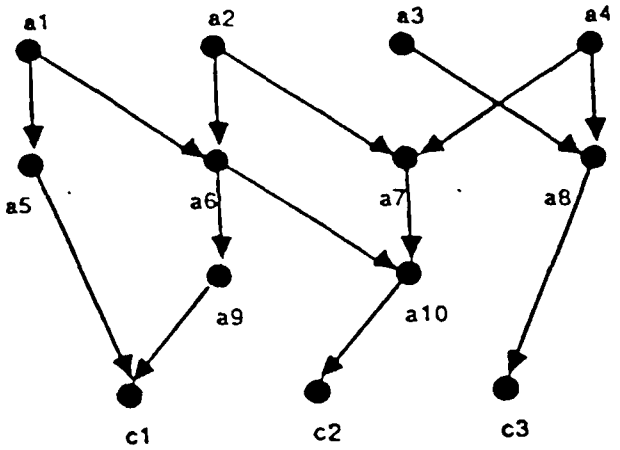


FIGURE 6. Production system with the OR of several paths to the same consequent c_n.

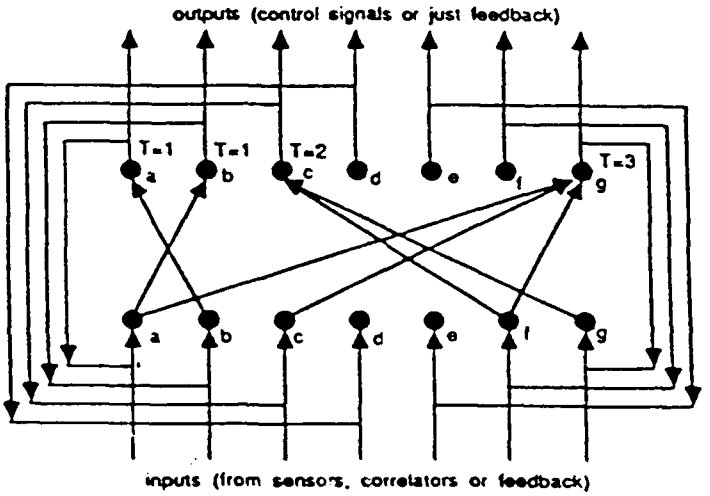


FIGURE 7. Neural net for the rules in Figure 5.

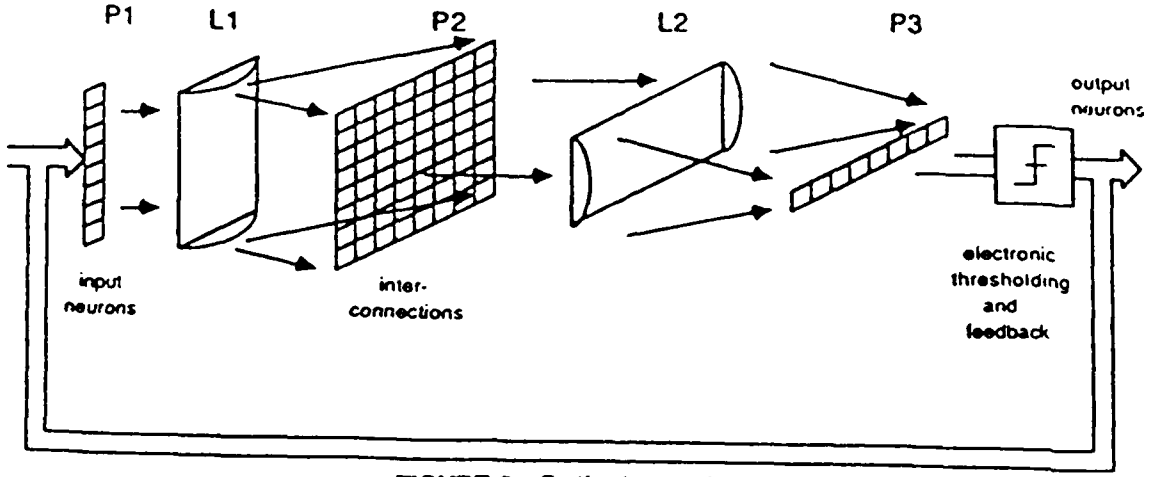


FIGURE 8. Optical neural net.

produced from generic shapes (object parts) using a Synthavision system including lighting and illumination effects. We divided the 9 objects into 3 clusters (Table 2) with various generic shapes (object parts) comprising each (Table 3). For each object and each of the 12 generic shapes, we generated from 3 to 5 different aspect views. Distortion-invariant projection SDF filters [2] were formed for each of the 12 shapes and tested against the distorted objects to verify that we could recognize each of the 12 shapes independent of distortions. The shapes within the test objects differed from those from which the filters were formed due to shading and illumination, edge thicknesses, and occlusion (e.g. the rods and posts on the fences, the wheels on the vehicles, etc.). Initial simulation results [4] were successful. In simulation tests, we showed success in recognizing distorted parts and discriminating between parts in objects. We now consider optical laboratory results obtained with a correlator.

c_{fire} = fire hydrant	c_{fence} = fence	c_{lamp} = street lamp
c_{tlight} = traffic light	c_{sign} = street sign	c_{car} = car
c_{truck} = truck	c_{house} = house	c_{motorc} = motor cycle

TABLE 1: Database of 9 Objects

CLUSTER 1:	SHORT OBJECTS (fire hydrant, fence)
CLUSTER 2:	TALL OBJECTS (traffic light, street sign and lamp)
CLUSTER 3:	BIG OBJECTS (motor cycle, car, house and truck)

TABLE 2: Multiple Object Clusters Used for First Separation of Objects

CLUSTER-1 PARTS:	short fat post, dome, horizontal bar
CLUSTER-2 PARTS:	long thin post, box, light, rectangular name plate
CLUSTER-3 PARTS:	gas tank, car body, wheel, big body, wedge shaped roof

TABLE 3: Symbolic Parts for each Object Cluster

5.2 Filter Synthesis and Testing

Projection SDFs [2] were formed with no false class training images used and with true class peaks set to 1.0. When a part was aspect view symmetric (e.g. the short fat post, the dome, the post and light), only one training image was used (otherwise the matrix used in synthesis is singular). For objects with more than one occurrence of one part, only one was used. The parts used were isolated and one fixed illumination was used for all aspect views of it. A 3×3 Sobel was used to edge enhance the data, which was then binarized to produce edges thicker than 2 pixels. In tests, different illuminations and edge widths plus occlusions occurred and when multiple parts are present in an

object, each has a different shape and edge width. In addition, in training the parts are rotated about their geometric center, while in testing the objects are rotated about a different point. Thus, considerable differences exist in the training and test data.

5.3 Advanced Considerations

When multiple parts are present in an object, the number of parts and their locations are useful features. All such parts cannot be located without reducing the threshold and allowing false alarms. The NN could possibly solve such problems. Increasing the number of training images to include the different versions of a part in each object would help, so would including false class training images. Another issue is the presence of several correlation peaks above threshold around a central peak (due to the various lines in the parts and object. Blob coloring allows us to select only the true central peak and will aid in locating multiple parts.

5.4 Resolution

We reduced the resolution for the street lamp and traffic light objects and the light part (in the street lamp) and the tbox part (in the traffic light) from 256×256 down to 32×32. Table 4 lists the autocorrelation peak (it is underlined) for the different inputs (horizontal) with the 2 different filters (vertical). All intra-image peaks (within the same true correlation plane with the part present in the object) were less than the autocorrelation peak value. The largest inter-image crosscorrelation peak value is given in the table (this is the largest peak anywhere when the input object does not contain the part). Table 4 shows P_C = 100% discrimination is possible with 64×64 resolution. The minimum true peak values remain about the same as resolution decreases, while the maximum false peak value increases. This is expected, since with reduced resolution objects look more similar and we expect both true and false peaks to tend to the same value.

5.5 Filter Quantization

We also quantized the number of amplitude levels in the image version of the filer. This is a practical issue in an optical realization. Table 5 shows selected results. No significant degradation in true or false peak values occurred down to 8 levels. The auto and crosscorrelation peak values with MSFs of the light and post are not affected as they use only one binary training image. As few as 4 levels can thus be used in encoding the filters.

5.6 Optical Laboratory Single Filters

All optical laboratory tests used a standard VanderLugt correlator with film input and the MSFs recorded on an NRC thermoplastic camera. The input FT lens L_1 had f_L = 495 mm, the FT lens L_2 had f_L = 400 mm and the correlation was determined with a camera. The reference beam angle used was θ = 30°, corresponding to a spatial frequency $\alpha = (\sin\theta)/\lambda = 8\times10^5$ cy/cm with He-Ne light (the center of the thermoplastic camera's bandpass response). The bias exposure for the thermoplastic camera was 20 μJ. The K ratio was 4-10 measured over the full reference-to-signal beams.

	lamp	tlight			
	0°	0°	20°	40°	60°
256 × 256 images and SDFs					
light	1.000	0.386	0.514	0.475	0.402
tbox	0.692	1.063	1.049	1.046	0.903
128 × 128 images and SDFs					
light	1.000	0.482	0.729	0.541	0.541
tbox	0.701	1.072	1.063	1.012	0.978
64 × 64 images and SDFs					
light	1.000	0.633	0.800	0.633	0.800
tbox	0.742	1.016	1.012	1.033	1.021
32 × 32 images and SDFs					
light	1.000	1.000	1.000	1.000	1.000
tbox	0.875	1.000	1.000	1.000	0.938

TABLE 4: Cluster 2 auto- and inter-image correlation peak values in spatial resolution experiments.

	lamp	sign				tlight			
	0°	0°	20°	40°	60°	0°	20°	40°	60°
Continuous levels									
name	0.747	0.995	1.008	1.069	1.106	0.625	0.704	0.670	0.613
light	1.000	0.440	0.425	0.394	0.386	0.386	0.514	0.475	0.402
tbox	0.692	0.682	0.679	0.686	0.682	1.063	1.049	1.046	0.903
post	0.683	0.667	0.667	0.667	0.667	1.000	0.989	0.989	0.989
256 levels									
name	0.747	0.999	1.009	1.069	1.106	0.625	0.704	0.671	0.612
light	1.000	0.440	0.425	0.394	0.386	0.386	0.514	0.475	0.402
tbox	0.690	0.680	0.676	0.683	0.680	1.060	1.045	1.044	0.899
post	0.683	0.667	0.667	0.667	0.667	1.000	0.989	0.989	0.989
4 levels									
name	0.805	1.132	1.001	1.155	1.215	0.668	0.751	0.734	0.725
light	1.000	0.440	0.425	0.394	0.386	0.386	0.514	0.475	0.402
tbox	0.691	0.706	0.710	0.715	0.709	1.078	1.060	1.056	0.876
post	0.683	0.667	0.667	0.667	0.667	1.000	0.989	0.989	0.989
2 levels									
name	0.784	0.722	0.802	0.989	1.342	0.737	0.811	0.802	0.722
light	1.000	0.440	0.425	0.394	0.386	0.386	0.514	0.475	0.402
tbox	0.652	0.596	0.621	0.621	0.583	1.200	1.022	0.924	0.972
post	0.683	0.667	0.667	0.667	0.667	1.000	0.989	0.989	0.989

TABLE 5: Cluster 2 auto- and inter-image correlation peak values in amplitude quantization experiments.

Table 6 shows optical laboratory results obtained for different SDF filters versus different input objects. The true peak values are for autocorrelation peaks. The largest false peak value anywhere is listed when a false class input is present. Tests (a) show that the post can be determined in all traffic light inputs. The post is also present in the sign and lamp objects and its peak values with these inputs are similar. The tbox in test (b) is only present in the tlight. It is recognized in all tlight inputs. It is not present in the sign and lamp objects and these peaks are all below the lowest true peak. Tests (c) and (d) show that multiple parts can also be recognized (in some object views, where expected).

Figure 9 shows various laboratory results. In Figure 9a, the input was the 0° light and the post filter was used. The 3 correlation peaks for the sfpost filter and the 0° fence are shown (Figure 9b) as are the four correlation peaks of the wheel and the 0° truck. The values in parentheses indicate the peak value (or their range for the case when multiple parts are present).

5.7 Optical Laboratory Frequency-Multiplexed Filter Tests

A frequency-multiplexed optical laboratory system was assembled. Four frequency-multiplexed SDFs were fabricated. Figure 10 shows the images of the 4 SDFs (two use only one reference pattern). These 4 SDFs were placed side-by-side in the input with 4 mm between each. The full input

(a) POST FILTER		(b) TBOX FILTER			
INPUT	TRUE PEAK	TRUE INPUT	TRUE PEAK	FALSE INPUTS	LARGEST PEAK
tlight 0°	121	tlight 0°	237	Sign 0°	115
tlight 20°	148	tlight 20°	227	Sign 20°	120
tlight 40°	109	tlight 40°	160	Sign 40°	143
tlight 60°	102	tlight 60°	158	Sign 60°	120
				Lamp	155

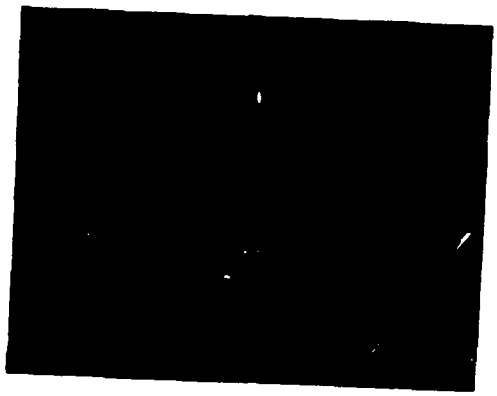
(c) SF-POST FILTER vs. 0° FENCE INPUT		(d) WHEEL FILTER vs. 0° TRUCK INPUT	
POST	TRUE PEAK	WHEEL	TRUE PEAK
1	111	1	234
2	96	2	255
3	103	3	216
		4	197

TABLE 6: Single Filter Optical Laboratory Test Results

was 14 mm wide. The frequency-multiplexed filter was formed with one exposure of this input. The output contains 4 correlation planes (left to right) of the input test object with filters of the post, tbox, light and name plate respectively. Figure 11 shows typical results obtained with 3 different test inputs.

5.8 Optical Laboratory CGH Results

Optical filters were also synthesized with a new CGH encoding technique [5]. The results are summarized in Table 7. The light part is present only in the lamp and the tbox part is present only in the tlight. They show much better agreement with simulations, since the CGH encoding used is very accurate.



(a) Correlation of the 0° tlight and post (121)

(b) Three correlation peaks for sfpost and 0° fence (96-111)

(c) Four correlation peaks for wheel and 0° truck (197-234)

FIGURE 9. Optical outputs of single filters.

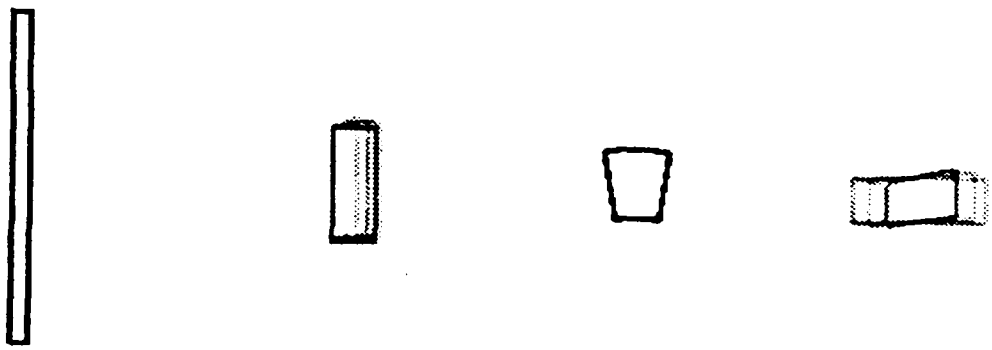
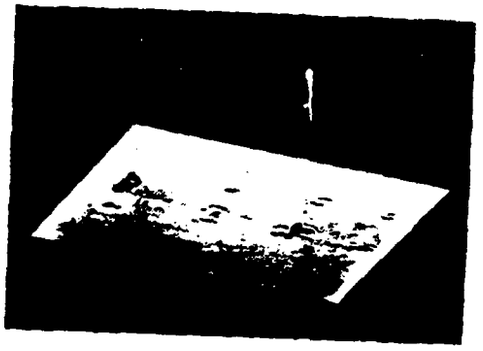
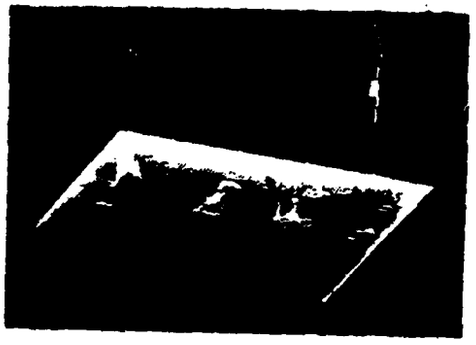


FIGURE 10. Multiple filter images for frequency-multiplexed filters.

(a) tlight input (contains post and tbox)



(b) lamp input (contains post and light)



(c) sign input (contains post and name plate)

FIGURE 11. Optical outputs of frequency-multiplexed filters of (left to right) the post, tbox, light, and name plate.

INPUT	lamp 0°	tlight 0°	tlight 20°	tlight 40°	tlight 60°	auto
light filter	64	41	51	41	51	64
tbox filter	45	62	61.5	63	62	61

TABLE 7: Optical Laboratory Results using New CGH Encoded Filters (64×64 images). The maximum false class peak anywhere and the true peak (underlined) are shown.

6. NEURAL NET PRODUCTION SYSTEM

We used the results of Section 5 for our parts list and our object tests, and our production system concept (Section 4) to devise a parallel (Figure 12) and an iterative (Figure 13) neural net production system design. We tested these neural net production systems on our database with successful results.

Acknowledgment

We gratefully acknowledge the support of the Defense Advanced Research Projects Agency monitored by the U.S. Army Missile Command.

REFERENCES

1. D. Casasent, "Optical AI Symbolic Correlators: Architecture and Filter Considerations", Proc. SPIE, Vol. 625, pp. 220-225, January 1986.
2. D. Casasent, "Unified Synthetic Discriminant Function Computational Formulation", Applied Optics, Vol. 23, pp. 1620-1627, May 1984.
3. E. Botha, D. Casasent and E. Barnard, "Optical Production Systems Using Neural Networks and Symbolic Substitution", Applied Optics, Vol. 27, pp. 5185-5193, 15 December 1988.
4. D. Casasent and E. Botha, "A Symbolic Neural Net Production System: Obstacle Avoidance, Navigation, Shift-Invariance and Multiple Objects", Proc. SPIE, Vol. 1195, November 1989.
5. P. Vermeulen, E. Barnard and D. Casasent, "New Fresnel CGHs for Lensless Optical Systems and their Applications", Proc. SPIE, Vol. 1052, pp. 223-233, January 1989.

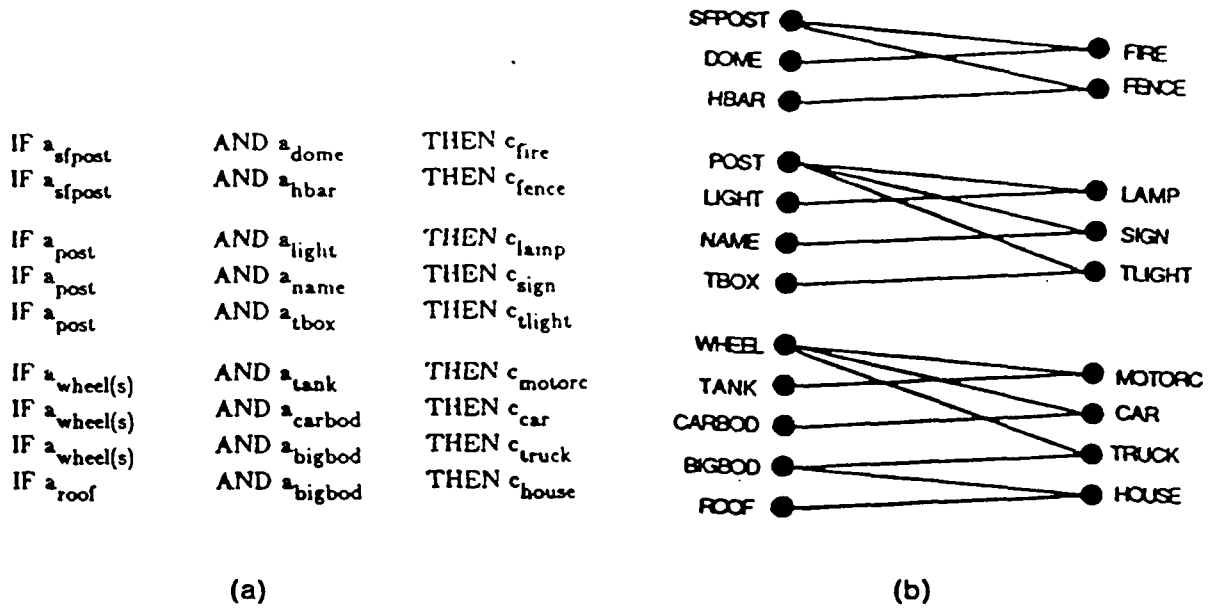


Figure 12. Rule base (a) and neural net (b) for a parallel production system.

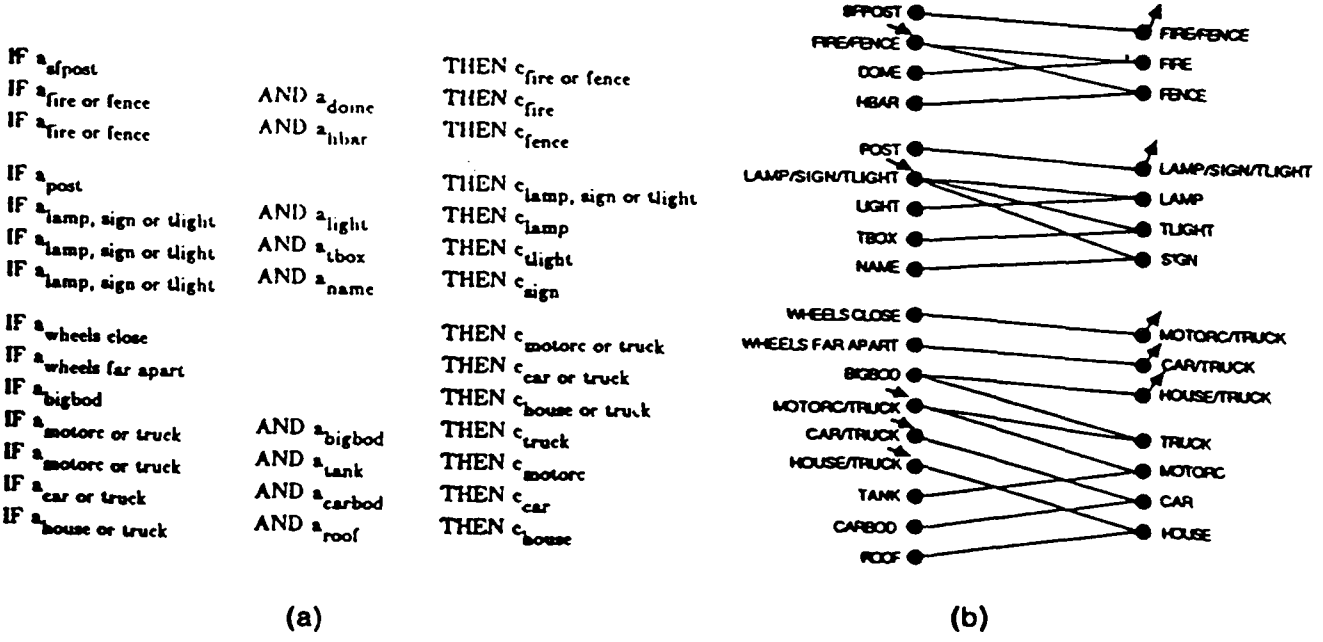


Figure 13. Rule base (a) and neural net (b) for an iterative production system.

CHAPTER 9

"Optical Production Systems Using Neural Networks and Symbolic Substitution"

Reprinted from Applied Optics, Vol. 27, page 5185, December 15, 1988

Optical production systems using neural networks and symbolic substitution

Elizabeth Botha, David Casasent, and Etienne Barnard

Two optical implementations of production systems are advanced. The production systems operate on a knowledge base where facts and rules are encoded as formulas in propositional calculus. The first implementation is a binary neural network. An analog neural network is used to include reasoning with uncertainties. The second implementation uses a new optical symbolic substitution correlator. This implementation is useful when a set of similar situations has to be handled in parallel on one processor.

I. Introduction

Several attempts have been made to apply optics to the field of symbolic computation. In one case[1] a hybrid optical and electronic architecture was proposed to implement a subset of the language PROLOG. The time-consuming parts of the PROLOG implementation are done optically, but extensive electronic control is employed and multiplexed filters which we use were not employed. This system is query- or goal-driven, while our production system is data-driven. This is not the general inference machine (production system) we consider, but serves as a coprocessor to a general-purpose electronic computer. An optical inference machine has been suggested[2] in which different situations are represented as adjacency matrices (these concepts arise from directed graph theory) and inferences are made by Boolean matrix addition and multiplication. This system can handle only the simplest case of Boolean logic, namely, $a \rightarrow b$ (a implies b) and its extension to handle subsequent inferences (which we consider) is not addressed. Architectures to extend the storage capacity (the size of a fixed knowledge base) of such systems using multiple holograms[3] have been described. Another production system[4] calculates probabilities associated with different inferences and assumes that the assertions in the knowledge base are independent events and that a linear relationship (a matrix) exists between the assertions (the input vectors) and the consequents (output vectors). This allows inferences to be made by a single matrix–vector multiplication followed by thresholding. Another similar expert or production system[5] allows the knowledge base to be updated during learning and uses the Bayes theorem to update the probabilities of consequents and rules and again uses optical matrix–vector multiplication to make inferences. We find the assumptions of independent events or assertions (which allows multiplication of probabilities when using the Bayes decision theorem) and mutually exclusive events (allowing the addition of probabilities) unrealistic in the situations where these systems are used (such as in medical diagnosis and vehicle control).

In this paper we show how we can use optics to implement a production system using two different approaches: the first employs an artificial neural network structure and the second is based on symbolic substitution. The neural network architecture employed is the well-known matrix–vector multiplication followed by a nonlinear function and feedback. Neural network architectures have not explicitly been used as iterative inference machines. However, Anderson[6] formulated an autoassociative memory in which each of the inference rules (if-then-rules) is encoded as one key/recall vector, with these vectors stored in an associative memory using the conventional outer-product formulation. When this system is presented with a partial input vector (only the antecedent elements), it iterates until the full vector is reconstructed (i.e., the consequent part is obtained). This system differs from ours, since it iterates to invoke one rule and no subsequent rules are examined. Thus, our use of a matrix–vector neural network as an iterative inference engine is new. The matrix and vector can be partitioned into facts and rules for different nearly uncou-

The authors are with Carnegie Mellon University, Department of Electrical & Computer Engineering, Center for Excellence in Optical Data Processing, Pittsburgh, Pennsylvania 15213.

Received 4 April 1988.
0003-6935/88/245185-09$02.00/0.

pled situations associated with a general problem. In this sense, the system is modular. This aids learning, since the system need not know and thus make all inferences in one pass through the system (as is required in a noniterative system). Rather, some inferences are made in each cycle and then other inferences are made on subsequent cycles using recently substantiated inferences. The nonlinearity present allows the system to model interdependent rules (that depend on several input facts; e.g., if two wheels and a rectangle are present, the object is a car; if two wheels and a person are present, it is a bicycle). In the prior examples, there is a large increase in confidence when the evidence that *a rectangle or a person is present* is included and this is modeled by the nonlinearity in the system. The nonlinearity thus allows the system to handle such situations, where the combination of the initial pieces of evidence is greater than the sum of their individual contributions.

Both of the systems we consider use a local representation of facts (one pixel per fact) and are thus less fault tolerant than a distributed representation (each fact is distributed to several inputs), since the loss of one input pixel will completely eliminate one fact. We also only consider the optical realization of a fixed set of rules that are known *a priori*. This allows the use of fixed filters and interconnection masks, which is presently more realistic than optical systems requiring adaptive filters and interconnections. However, our system allows new rules to be inferred that are not explicitly encoded.

In Sec. II we define our terms and the initial scenario addressed. In Sec. III we describe a binary neural architecture that can handle only binary (yes/no) type decisions and an analog neural system that includes the calculation of probabilities (without assuming mutually exclusive events and/or independent assertions). Section IV describes a symbolic substitution production system and a scenario where several similar parallel situations occur. In this case, a symbolic substitution implementation is preferable. We conclude with a discussion in Sec. V.

II. Definition of Terms and Scenario

In artificial intelligence, a production system consists of data, operations, and control.[7] In real systems, these distinctions often become fuzzy. In the production system we consider, the data are referred to as a knowledge base, consisting of a set of facts and a set of inference rules (also called production rules), that constitutes the operations performed on the facts. The system control handles the order in which the rules are invoked. In a data-driven system, the control selects the rules to be tested. These are the rules associated with present facts/data. This process continues until the goals are satisfied. In a goal-driven system, the control first selects the rules associated with the goals and proceeds with the inferences to arrive at the present facts/data (if possible). In the applications we have in mind, we refer to the production system as an inference engine, an expert system, or a rule-based system, without any fine discrimination between these terms.

We choose propositional calculus as the formalism in which to represent the knowledge base of the production system. Propositional calculus is a subset of predicate calculus, which is a formal language. The elementary components of predicate calculus[7] are symbols that are combined to form what are called atomic formulas. Sentences or expressions in predicate calculus consist of connected atomic formulas and are called well-formed formulas (wffs). In predicate calculus, expressions can contain variable (among other) symbols, four connectives, and quantification is allowed. Propositional calculus allows the use of the same four connectives between atomic formulas as predicate calculus, namely, AND denoted by $\wedge$ (which forms a conjunction), OR denoted by $\vee$ (forming a disjunction), IMPLIES denoted by $\rightarrow$ (which forms an implication), and NOT denoted by $\sim$ (forming a negation). The connection (i.e., the result of applying any of the connectives: conjunction, disjunction, negation, or implication) of any number of wffs is also a wff. In an implication, the left-hand side is called the antecedent and the right-hand side the consequent. Atomic formulas in propositional calculus contain only predicate, function, and constant symbols (the use of variable symbols such as x is precluded). To illustrate the meaning of these symbols, consider the fact "Dave's student wrote a paper." This sentence would be represented as the atomic formula WROTE [PAPER, student(DAVE)]. "WROTE" is the predicate symbol that represents a relationship; "PAPER" and "DAVE" are constant symbols and represent objects or entities; "student" is a function symbol that maps constant objects or entities to one another. Propositional calculus was chosen rather than predicate calculus since its operations can be realized more easily on the architectures we consider. We will refer to the atomic formulas as "facts" and the implications (containing any of the symbols and connectives) as "rules" in the knowledge base we discuss.

The types of implications or rules we consider are called if-then-rules. Our knowledge base contains a set of statements (assertions or facts) that are true at any specific point in time and a set of if-then-rules that are true at all times. The facts represent the current world state and are considered the explicit declarative knowledge of the system. The rules encode the general knowledge about the world and are the implicit declarative knowledge. The production system makes inferences based on both the facts and the rules. If any new assertions become true during one cycle of the inference process, they are included as facts in the input to the system during the next inference cycle.

The production system we discuss is a forward-chaining or data-driven system, as opposed to a backward-chaining production system that is goal-driven. In any production system, a set of goal formulas is defined. When any one of the goal formulas becomes true, the system has reached its final state and the inferences stop. In a forward system, the initial state

of the system is set to the facts in the database that are true and the system is allowed to make inferences until one of the goal formulas becomes true.

We restrict the rules in our knowledge base to be of the form

For rule r:

IF $(a_m \text{ AND } \ldots \text{ AND } a_n)$
OR $(a_p \text{ AND } \ldots \text{ AND } a_q)$ (1)
OR ...
THEN c_r

where the facts a_i can be negations and where the sets $\{a_m \text{ to } a_n\}$, $\{a_p \text{ to } a_q\}$, etc. are not necessarily disjunct (to allow the OR statement description above). The set elements a_m, etc. are the antecedents for rule r and c_r is the consequent. An example of a knowledge base in a forward production system is shown in the graph or decision net in Fig. 1. Every possible fact a_i or c_r in the knowledge base is represented by at least one node (a_1 to a_{10} and c_1 to c_3), although Fig. 1 shows only one node per fact. The consequents of the rules are represented by the bottom nodes (c_1 to c_3) and the antecedents by all the other nodes above them (a_1 to a_{10}). A specific rule r is thus represented by the OR of the different paths to a consequent node c_r with each path being the AND of a set of antecedents as in Eq. (1). For the example in Fig. 1, rule 1 is

IF $(a_1 \text{ AND } a_5)$
OR $(a_1 \text{ AND } a_6 \text{ AND } a_9)$
OR $(a_2 \text{ AND } a_6 \text{ AND } a_9)$
THEN c_1.

As Fig. 1 shows, our knowledge base with rules in the form of (1) will not have a tree structure. Rather, as in Fig. 1, there may be several root nodes that lead to the same intermediate node (e.g., a_2 and a_4 can both lead to the intermediate node a_7), and intermediate nodes that lead to one another (e.g., a_6 to a_{10}) and/or to the same bottom nodes (e.g., a_5 and a_9 lead to c_1). The knowledge base in Fig. 1 is completely described by the three rules for the three consequents. Thus it can be implemented in one cycle (matrix–vector multiplication). However, if $c_1 = a_3$, the iterative nature of our production system and its ability to make many new inferences from a limited number of facts and rules can be seen. Specifically, if c_1 is true and $c_1 = a_3$, the system learns other rules (e.g., a_1 AND a_5 AND a_3 AND $a_8 \rightarrow c_3$, a_1 AND a_6 AND a_9 AND a_3 AND $a_8 \rightarrow c_3$ and a_2 AND a_6 AND a_9 AND a_3 AND $a_8 \rightarrow c_3$) on successive iterations (without having been told these rules explicitly).

As a specific example of our production system, we chose the problem of controlling a mobile robot (such as the autonomous land vehicle[8]) in an urban area. This example will allow us to detail specific rules (in terms of objects such as cars rather than abstract elements such as a_n and c_r) and to specify when a neural or symbolic substitution production system is appropriate (as we will detail). This is a case in which most of us are experts due to our experience in driving motor vehicles. The final results of the inferences (the goal formulas) are control instructions such as "slow down," "blow the horn," etc. We assume that the vehicle is equipped with a Doppler radar that provides velocity information and a video camera looking in the same direction as a human driver of a vehicle usually does. The video images are first processed by a symbolic correlator as described elsewhere[9] and summarized below.

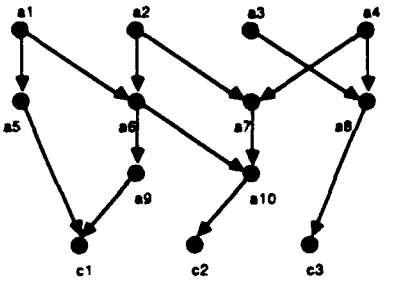


Fig. 1. Example of a decision net representation of a rule base.

The symbolic correlator compares the image with a set of geometric symbols of various forms, scales, and rotations and outputs a symbolic description of the objects in the image and their spatial locations. This denotes which geometric symbols are present, their scales, rotations, and relative positions. This correlator is part of a symbolic preprocessing system that has its own knowledge base and that infers the identity of the objects in the field of view. The stored geometric symbols are the facts in the symbolic correlator's knowledge base. Geometric models of the possible objects in terms of geometric symbols are the rules in the knowledge base. The preprocesssing symbolic correlator system draws conclusions on the identity of the object based on the geometric model rules. The emphasis in this paper is on the production system that operates on the symbolic correlation object output data and not on how the symbolic correlation data are obtained (although an analog production system could be used to obtain this data).

We assume three categories of facts in our production system:

(1) inputs from the symbolic correlator (e.g., that the obstacle in the field of view is a car, plus its position and orientation);

(2) inputs such as the velocity of the objects; and

(3) control and monitoring signals from the vehicle's steering and acceleration subsystems (that give the speed of the vehicle and its direction of travel).

The rules in our production system include knowledge about how to steer and control the vehicle in the presence of obstacles in different situations. As an example of a typical situation, consider the case when the vehicle is moving along smoothly in the right lane of the road (which is indicated by a curb, i.e., a solid line and some elevation, or by gravel, i.e., a change in texture on the right side and a broken white line on the left side). A typical situation would be when another car suddenly pulls out in front of the vehicle and moves slower than the vehicle. In such a situation, appropriate control signals or consequents of goal formulas associated with this situation could be to blow the horn, change lanes, slow down, stop, etc. The rules are

such that they use the specific symbolic preprocessor outputs, the available sensors, and the control and monitoring signals, e.g.,

IF (object in field of view is a car
AND on collision course with this object
AND object moving slower than vehicle)
THEN slow down.

III. Neural Network Architectures for Production Systems

We first consider the use of a synthetic neural network to implement the production system. As discussed, we use a forward-chaining system, i.e., a data-driven production system. Such a system does not know in advance which specific goal(s) to look for, and thus it bases its inferences on its current known state of the world and not on what other facts are necessary to satisfy a particular goal.

A. Binary Neural Network System

As an initial case, we consider binary-valued decisions and logic statements, i.e., we do not consider reasoning with uncertainties. We use a two-layer neural network with binary neurons (neurons with outputs that can take on only the values 0 and 1) to store the rules and make the inferences. The input nodes (first layer) represent the antecedents of the rules and the output nodes (second layer) represent the consequents. The antecedent nodes in the input are connected to their consequent nodes in the output by links of weight one (since uncertainties are not considered). The links store the connectives (in our example, they are only ANDs) of the rules. The output nodes are nonlinear computing elements that sum their inputs and threshold the result to give binary-valued outputs. In a forward-chaining system, the initial outputs (on the first iteration) prove certain consequents to be true. These consequents then become facts or assertions that are input to the system (by feedback) on the second iteration. Thus, there must be an input and output node for every possible assertion (all antecedents and consequents) in this production system.

The neural production system is best seen by an example. Figure 2 shows the interconnection pattern for the neural network for the example in Fig. 3. This problem can be posed as a matrix–vector multiplication, if we assign all assertions (antecedents and consequents, i.e., a through g) to different binary input and output vector elements and if we describe the interconnections (the left-hand side expressions in Fig. 3) as binary 2-D matrix elements. The matrix–vector multiplication and subsequent thresholding then yields the output vector of consequents. The output threshold (denoted by T in Fig. 2) varies from element to element, depending on the number of antecedents in the corresponding rule. By replicating several connections, all output neurons can be made to have an equal number of inputs and thus a uniform output threshold can be used and all interconnections still have the same strength. This approach circumvents

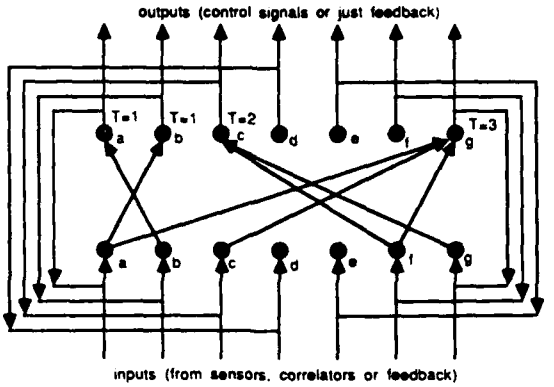


Fig. 2. Neural network for the rules in the example in Fig. 3.

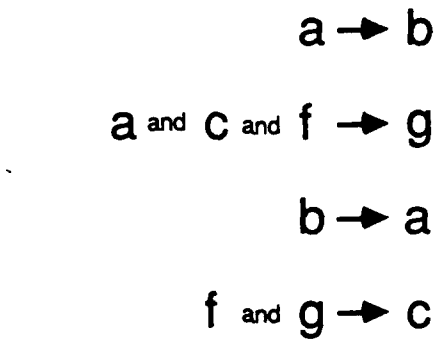


Fig. 3. Rules for the neural network example in Fig. 2.

the need that other neural network architectures have of requiring varying thresholds or nonuniform interconnection strengths. The antecedent (input) and consequent (output) vectors together constitute the current world state, which is a list of all the true assertions at a given point in time, i.e., after one pass through the system. In the example in Fig. 2, if, at cycle or iteration n, the vector elements representing antecedents a, c, and f are all 1, the vector element representing the consequent g will be 1 after multiplication and thresholding. The resulting consequent vector after cycle n could be logically ORed with the prior input vector, to provide the new input for the next pass (cycle $n + 1$) through the system. The new input thus has all the new consequents obtained in pass n included as facts or true antecedents for cycle $n + 1$ plus the old true antecedents. This is necessary, since some of the consequents from one pass can be antecedents for rules and thus new consequents may be satisfied on a subsequent pass. We can simplify this system and avoid the logical OR function by connecting each input with its corresponding output neuron with an interconnection strength equal to the output threshold required or with multiple unit-weight interconnections which equal the strength required to fire the output neuron. This ensures that once a fact (neuron) has been proved (fired), it remains true. This is equivalent to having unit diagonal elements in the interconnection matrix. With unit diagonal elements, conventional neural networks such as the Hop-

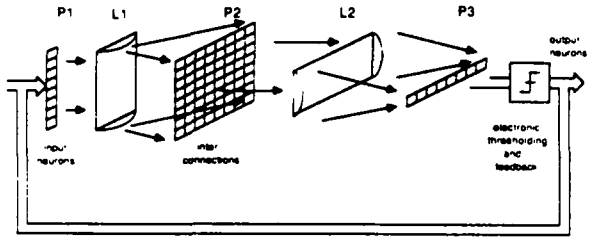


Fig. 4. Schematic of an optical matrix–vector implementation of the neural network in Fig. 2.

field model cannot be shown to converge. However, as noted in Sec. I, our production system application of neural networks is different and thus is not subject to the same constraint. This process (of matrix–vector multiplication, thresholding and ORing) is iterated until one of the goal formulas is proved or until no change occurs in the output neuron states.

We can conceive of situations where we want to deviate from the constraint of unit diagonal elements for all the facts. For example, a neuron that gets its input from the symbolic correlator output should only fire when the geometric symbol that it represents is present and should stop firing when that symbol disappears. Such a neuron should not have a connection between it and the output neuron correpsonding to it. However, a fact such as "the final destination is Little Rock, Arkansas" will only be true during initialization of the network and the network will have to "remember" it, i.e., the neuron representing it should keep firing. This can only be achieved by a unit diagonal element and feedback. Therefore, in the design stages, all the facts that have to remain true once they are fired should be assigned a unit diagonal element. As noted earlier, this is a new synthesis method for the connection matrix suitable for our application.

Figure 4 shows the schematic of an optical implementation of the neural net in Fig. 2. We use an optical binary vector–matrix multiplier with a 1-D array of laser diodes at $P1$ (for the input vector elements), a 2-D matrix of rules at $P2$, and a linear array of detectors at $P3$ (whose vector output is the matrix–vector product). After thresholding, these $P3$ outputs yield the neuron states (and the vector for the next iteration). The matrix elements (interconnections) are fixed (and binary) so that film can be used at $P2$. The thresholding of the $P3$ result can be performed electronically in the feedback loop to the input $P1$ laser diode array or can be included optically on the output $P3$ device. The output vector signals can be fed back electronically or optically with mirrors or optical fibers (in an all-optical system). Figure 4 indicates electronic thresholding and feedback. The optical connections are easily achieved. The light from $P1$ is spread out horizontally by cylindrical lens $L1$ to uniformly illuminate the rows of the 2-D array in $P2$. The array at $P2$ has the connections from $P1$ to $P2$ encoded on it as a matrix. The light leaving $P2$ is the point-by-point product of the input $P1$ vector and each of the column vectors in $P2$. These products are integrated (summed) vertically and imaged horizontally by cylindrical lens $L2$ onto the 1-D detector array in $P3$. The driver circuits for the input laser diodes could have memory (so that once a laser diode is on, it stays on; with nonzero diagonal interconnection matrix elements this is not necessary). With this system, all possible inferences based on the current world state are made in parallel. This produces a new world state on which all inferences are made in parallel, etc. If the spatial light modulator at $P2$ has 1000×1000 elements, the input $P1$ and output $P3$ vectors have 1000 elements and the capacity of the production system is 1000 assertions.

B. Analog Neural Network System (Including Probabilities)

We now consider the case of reasoning with uncertainties, i.e., when probabilities are associated with the antecedents and/or the consequents of the rules. The probabilities are heuristic and are derived from our general knowledge about the world. This case can be handled by an analog neural network. The input and output neurons (vector elements) are now analog (not binary as in the previous discussions) and can assume any real value between 0 and 1. Each neuron represents a fact (input antecedent or output consequent) and its output (the amount by which it is firing) is the probabilitiy of that fact being true. The connection strengths (matrix elements) between the input neurons (the antecedents of the rules) and the output neurons (consequents) represent the contributions of the antecedents to making the different consequents true. The outputs from the antecedent neurons are multiplied by the connection strengths and a nonlinear operation is performed on the sum at each of the output neurons. The nonlinear function can be a sigmoidal function $f(x) = 1/[1 + \exp(-\alpha x)]$, where α determines the slope of the nonlinearity. In the limits, the sigmoidal function can be a step ($\alpha = \infty$) or approximately linear when α is made very small compared to the inverse of x in the region of interest.

Let w_i be the probability of antecedent i being true (the amount of trueness of fact i) and let $p_j(w_i = 1)$ be the probability that consequent j is true given that antecedent i is true. For illustration, consider a rule in the knowledge base that infers that the object in the image is a truck if there are two circular and one rectangular geometrical symbols present. Let w_1 represent the probability that a circular symbol is present, let $w_{1,1}$ be the probability that two circular symbols are present, and let w_2 be the probability that a rectangular symbol is present. Also let p_1 be the probability that the object is a truck. Let us heuristically assign a probability of 0.1 that the object is a truck if there is one circular symbol present, a probability of 0.2 if only a rectangle is present, and a probability of 0.8 that the object is a truck if two circular symbols and a rectangle are all simultaneously present. In our notation, the probabilities are written as

$$p_1(w_1 = 1) = 0.1,$$

$$p_1(w_2 = 1) = 0.2$$

$$p_1(w_{1,1} = 1; w_2 = 1) = 0.8.$$

Note that $p_1(w_{1,1} = 1; w_2 = 1) = 0.8$ is not equal to $2p_1(w_1) + p_1(w_2) = 0.4$. Thus, the probability of a truck increases nonlinearly as more evidence is gained (i.e., the probability of a truck, when three of its geometrical components are present, is larger than the sum of the probabilities when each of the individual components is present). Thus a nonlinear function is needed to map the inputs to the proper output neuron strength.

Figure 5 shows the interconnection strengths and operations required for a general neural network for reasoning with uncertainties. The strengths of the output (consequent) and input (antecedent) neurons are given by c_i and a_i, respectively. The connection strength between a_i and c_j is denoted by w_{ij} and the strength of a given output element is $c_j = f(\Sigma\, w_{ij}a_i)$, where f denotes the output nonlinear function and the sum is over all assertions. Let us now detail how analog values might be assigned to the strength a_i of the input neurons. Each fact, such as "a circular symbol is present," is associated with one of the input neurons. If these facts are obtained from a symbolic correlator and if the correlation peak value is 0.6, we would assign a certainty of 60% to that statement and the output strength of this neuron would be 0.6. The inputs to a second-layer neuron are the products of the outputs of the input first-layer neurons and the weights that connect them to that output neuron. Each output neuron forms the sum of its inputs and performs the nonlinear operation f on the resultant sum. These outputs indicate the probabilities that the associated consequents are true. The slope of the nonlinear function and the weights connecting the input and output neurons are used to provide an output c_j equal to the proper analog probability. Since it is preferable to have the same nonlinear function for all output neurons, we fix the output nonlinear function and vary the interconnection weights w_{ij} to achieve c_j outputs equal to the desired probabilities. In this analog neural network, the optical mask at $P2$ of Fig. 4 must have analog transmittances.

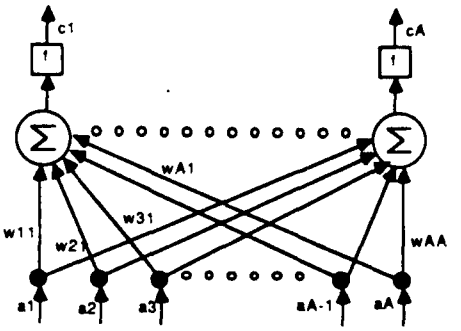


Fig. 5. Interconnection strengths and operations required in a neural network for reasoning with uncertainties.

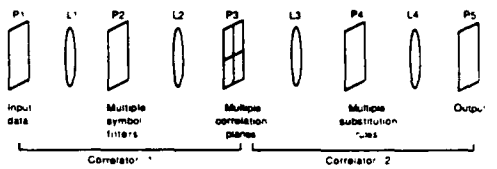


Fig. 6. Cascaded optical correlator system for symbolic substitution.

IV. Symbolic Substitution Production System

A forward-chaining production system can also be implemented using symbolic substitution. The use of a symbolic substitution production system is appropriate when a set of similar situations (such as the control of a fleet of vehicles) has to be handled in parallel on one processor (as we will detail later). Symbolic substitution[10] involves two steps: the recognition of an input symbol and the substitution of an output symbol for that input symbol. This pair of input–output symbols is called a substitution rule. The set of substitution rules specifies the symbolic substitution system. The implementation of symbolic substitution we consider uses two cascaded optical correlators as shown in Fig. 6. In this architecture, frequency-multiplexed matched spatial filters of the possible input symbols are placed in $P2$ and the first correlation plane $P3$ has peaks in different regions denoting where the different symbols are in the input plane $P1$. We threshold $P3$ to provide delta functions at the locations of the input symbols. $P3$ serves as input to the second correlator. The Fourier transforms of the output symbols to be substituted are spatially multiplexed at $P4$. This achieves the substitution of the output symbol by convolution with the delta functions. The different spatial frequency carriers at $P4$ produce the sum of the different substituted patterns at $P5$ as is desired. This system is detailed elsewhere.[11,12] The typical use for these systems has been in optically performing logic and numberic functions. We now consider its use in a new production system application.

A. System Overview

We first overview our symbolic substitution production system concept. For our production system application, the input to the symbolic substitution machine is a coded image representing the facts (or assertions) that are true at this point, with each possible assertion represented as a specific symbolic pattern in a specific location. An assertion is zero until it becomes true. The system implements the if-then-rules in the knowledge base by looking at which antecedents (symbols) are present in the input and substituting the symbols for the consequents of those rules whose antecedents are present. We substitute only for assertions that were not true previously and do not change symbols for assertions that were previously true since they are still true. By changing zero symbols the system thus adds the new facts (consequents) that become true at the invocation of the rules. As noted earlier, the consequents are placed at their proper locations in the output.

B. Filter Organization

Symbolic substitution logic and numeric processors (such as Fig. 6) substitute patterns in the same location as the input patterns. However, since our symbolic substitution production system substitutes output symbols in different locations from those of the input symbols, the second correlator in the cascade in Fig. 6 will be modified from its prior versions[11,12] as we discuss in Sec. IV.C. As an example, consider the case when input and output facts (i.e., antecedents and consequents) are the same (e.g., our $c_1 = a_3$ example in Fig. 1). In this instance, we assign the same location and symbol to both. If both a fact and its negation enter as antecedents a_n, each is assigned a separate location and symbol. We also allow separate locations for objects in different positions in a scene (e.g., the car is in front, back, left, or right in our mobile vehicle example).

We now discuss the specific organization of the filters. Our rules are in the form of the OR of antecedent sets (each of which is the AND of a set of antecedents), i.e.,

$$\begin{array}{lll} \text{IF} & & (a \text{ AND } b) \\ & \text{OR} & (c \text{ AND } d) \\ \text{THEN} & & x. \end{array} \tag{2}$$

In the example in (2), there are several antecedent sets (which we refer to as conjunctions). Each of these must be viewed as a separate filter at $P2$ of Fig. 6, each conjunction has a separate location assigned to it in $P3$, and all conjunctions associated with the same consequent are assigned to the same region (output correlation plane) in $P3$. The filter implements the logic AND operation directly (rather than using the symbolic substitution system to implement logic functions). We now detail these issues. The rule in (2) can be rewritten as two rules with the same consequent

$$\begin{array}{l} a \text{ AND } b \rightarrow x \\ c \text{ AND } d \rightarrow x. \end{array} \tag{3}$$

Each rule is represented by a separate filter function with the filters being the conjunctions a AND b and c AND d. Thus, the two filters each contain both antecedents and their locations properly encoded and the logic AND operation is included in the filter design. To see why separate filters and $P3$ locations are necessary for each conjunction, recall that several antecedents exist within each conjunction and several conjunctions yield the same consequent as in (2) and (3). If the same $P3$ location were assigned to the two conjunctions in (2) or (3), then if only the antecedents a and c (i.e., parts of each of the two conjunctions) were present, the $P3$ output would exceed threshold (by summing partial correlations from several conjunctions).

The organization of the rules as the OR of antecedent sets of AND operations was chosen since it is computable with the production system implementation described above. The optical filters perform the AND (and hence the conjunction operation) by addition on the $P3$ thresholding array device. We arrange the location of the output correlation peaks such that conjunctions with the same consequent lie in different locations in the same region of $P3$. If a peak exceeds threshold anywhere within one of these $P3$ regions, the second correlator ($P3$ to $P5$) substitutes the proper consequent symbol in its proper location in $P5$. Once the symbolic substitution has been done at $P5$ for one pass through the system, the $P5$ outputs are fed back and ORed with the prior $P1$ data to produce the new antecedents that are now true for the next pass through the system. Thus, this architecture also iterates similarly to the neural network system. We now detail the filter design, symbolic patterns used, and a new optical architectural implementation.

C. Filter Implementation and Symbolic Pattern Selection

Since the rules (separate conjunctions are separate rules) are not linear, we cannot implement the symbolic substitution in one correlator, rather we require a cascaded correlator such as in Fig. 6. The first correlator ($P1$ to $P3$) achieves the recognition and conjunction of each antecedent set and each conjunction has a specific location in $P3$ assigned to it (i.e., a correlation peak will appear at a specific location in $P3$ if a given conjunction is true). This is easily achieved since the location of each antecedent in $P1$ is known, fixed, and specified. Thus the $P2$ filters include this positional $P1$ information and the carrier spatial frequency for each filter is chosen to select the $P3$ region of the correlation peak. The specific location of the correlation peak in the region (output correlation plane) is determined by the correlator. We select the positions of the antecedents in $P1$ such that antecedents that are members of conjunctions associated with the same consequent lie in the same region in $P3$. This reduces the range of spatial frequencies required on the $P4$ filters (discussed later) to read out the same consequent pattern for each conjunction in the same region of $P3$. Since the locations of the antecedents in $P1$ are known, the locations of the correlation peaks in $P3$ can be specified. The $P2$ filters can be spatially or frequency-multiplexed or a combination of spatially and frequency-multiplexed $P2$ filters can be employed to achieve the required results. The substitution filters at $P4$ (in the second correlator) are required to substitute (activate) the symbol (in a specified location in $P5$) for any consequent that is now true or instantiated (given the present world state of antecedents). Since a peak (after $P3$ thresholding) anywhere in any of the $P3$ regions (corresponding to different consequents) should activate the associated consequent symbol, we encode the different consequent symbol patterns on different sets of spatial frequencies at $P4$. The spatial frequencies used for the filters of one consequent are determined from the positions in $P3$ of correlation peaks (corresponding to different conjunctions that yield the same consequent). Thus, a correlation peak in any of a number of specified locations in $P3$ yields activation of the symbol for that consequent at the specified $P5$ location. The instantiated consequents are superimposed at $P5$ with the $P5$ threshold set to implement the OR function in rules of the form of (2).

The $P5$ data are then fed back to $P1$ to produce a new world state of antecedents for the next iteration. If the spatial light modulator at $P1$ has memory, the $P5$ data can be simply fed back iteratively to $P1$ to form the new antecedents (the OR of the prior facts and the new consequents/facts). This use of symbolic substitution differs from prior work (involving the implementation of logic and numeric functions) since the output symbols are substituted in different locations, not *in situ*.

We now consider the symbolic patterns used for the antecedents and consequents. If simple points in specified $P1$ locations (rather than patterns) are used to denote true antecedents, a set of input points with given relative spatial locations can instantiate an output $P3$ correlation peak (regardless of the absolute location of the set of points in $P1$). This will yield false peaks in $P3$ at locations proportional to the position of the set of points in $P1$. In this instance, this is a disadvantage of the use of a correlator when the locations of input facts are known (i.e., a correlator automatically searches all absolute locations of antecedent set patterns, even though this is not necessary here). The possible cross-correlation effect of this at $P3$ can be reduced by using symbolic patterns (not points) for each fact. Two choices exist for these symbolic patterns. We can use binary patterns (i.e., use $\log_2 N$ pixels to represent the patterns for N facts). This requires $N \log_2 N$ pixels in $P1$ for N facts (antecedents and consequents). As an alternative, we can use orthogonal patterns for the symbols for the N facts. This will remove cross correlations at the peak location at the cost of an increase in the space–bandwidth product (SBWP) required at $P1$. Specifically, the use of orthogonal symbolic patterns requires N pixels per symbol or N^2 pixels to encode N symbols (antecedents and consequents) compared with $\log_2 N$ pixels per symbol or $N \log_2 N$ pixels to encode N symbols with binary patterns. Thus, the use of binary symbols reduces the required SBWP at $P1$ by the ratio $(\log_2 N)/N$. For large N, this can be significant (e.g., for $N = 1000$, the use of orthogonal symbolic patterns increases the SBWP required at $P1$ by a factor of 100). Hence, the use of binary encoded symbolic patterns is preferable from practical implementation considerations. This seems to be realistic, since the probability is small of finding simultaneously: M instantiated antecedents (whose relative spatial locations are the same as that of one of the possible correct conjunctions) whose correlation with the $P2$ filter for that conjunction would yield a correlation peak in $P3$ at an allowed location, and that these antecedent patterns will each have a high cross correlation with the corresponding true set of antecedents. Thus, we adopt binary-encoded symbolic patterns and consider orthogonal patterns only when the device at $P1$ can accommodate the increased SBWP.

Two other issues associated with the symbolic pattern selection merit attention and discussion. Since the locations of possible correlation peaks in $P3$ are known, we are not concerned with cross-correlation values associated with the correlation of a reference and shifted versions of the input symbols [since we will utilize a mask of apertures (holes) in front of $P3$ with apertures placed only at the locations of possible legitimate conjunctions]. Other researchers considering symbolic substitution correlators have expressed concern about this. A second issue is the need for the different symbolic patterns to have the same energy (same number of black-and-white pixels). This is not essential, since the weight of the different filters can be adjusted to yield the same correlation peak height, even if the energy of the N symbolic patterns differs. However, when cross correlations are considered, their normalization with respect to all possible input patterns is not easily achieved. Thus, it is preferable for each symbolic input pattern to be the pattern plus its conjugate. This is detailed elsewhere[9,1] and thus an increase in the SBWP of symbolic patterns by a factor of 2 is required to include this effect. This also solves the cross-correlation problem at peak locations discussed earlier, since it assures that every conjunction now has a unique active (white pixels) symbolic pattern. In this case, the cross correlation of any two patterns will differ by at least $1/N$ from the autocorrelation and can thus be removed by thresholding.

D. Alternative Optical Production System Symbolic Substitution Architectures

Since the shift invariance of a correlator is not necessary, one could implement the production system with the $P1$ to $P3$ system being a matrix–matrix or vector inner product (VIP) multiplier. If symbolic patterns were not used for the $P1$ facts, such a system would be identical to the neural network architecture of Sec. III. The disadvantage of using a VIP processor is that multiple situations (i.e., control of a fleet of autonomous vehicles) cannot be easily handled and that the basic cascaded optical correlator architecture would be altered (and thus the same optical system would not allow multifunctional use for logic, numeric, morphological, and production system applications). To retain the ability of this system to handle multiple situations (e.g., a fleet of vehicles), we must retain the shift invariance of a correlator. However, with facts encoded on one of several lines in $P1$, we require only vertical shift invariance. Thus, the use of a 1-D vertical correlator would suffice and reduce cross-correlation peak intensities (since horizontal shifts of $P1$ patterns would now not contribute to false correlation peaks).

We now advance a preferable new optical symbolic substitution production system architecture (Fig. 7) for this case in which the location of all possible facts is fixed (position encoded). In this case, we use a binary spatial light modulator (SLM) at $P1$ with one pixel per fact. This significantly reduces the SBWP requirements for $P1$. We use the position encoding of $P1$ symbols to encode the symbols for each fact with a fixed *in situ* mask at $P2$ placed directly behind $P1$. Thus, we need only activate one pixel on the binary SLM at $P1$ and this automatically inputs a 2-D sym-

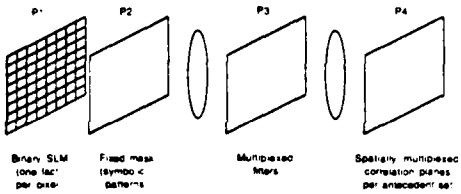


Fig. 7. New optical production system symbolic substitution architecture with reduced SBWP requirements and fixed symbol mask.

bolic pattern to the system (with the symbolic pattern provided by the fixed film $P2$ mask, thus reducing the SBWP requirements for realistic real-time SLMs at $P1$). This new architecture allows a larger production system (one with more facts) to be realized, without wasting real-time SBWP at $P1$ for the symbolic pattern encoding. The system of Fig. 7 also uses only one correlator, rather than a cascaded correlator. In this case, the feedback from $P3$ to $P1$ requires a simple CGH, fiber-optics connection, or electronic feedback system which activates a $P1$ pixel if any pixel in a given $P3$ region is activated. This new Fig. 7 architecture thus requires a single three-plane one-correlator system that is structurally similar (and no more complicated) to the neural network architecture of Fig. 4, but which allows multiple situation processing. The two neural network and symbolic substitution approaches are compared in Sec. V.

V. Discussion

Now that both this symbolic substitution implementation (Sec. IV) and the neural network realization (Sec. III) have been described, let us discuss when the symbolic substitution implementation is preferable.

In the original symbolic substitution system of Fig. 6, each fact is encoded as a symbolic pattern, whereas the neural network requires only one pixel per fact. Hence the SBWP requirement of the original symbolic substitution system is larger than that of the neural network. However, the new proposed optical symbolic substitution processor of Fig. 7 allows equal input SBWP requirements for both approaches. In addition, if the SBWP requirement is greater than the input 1-D SBWP, additional facts can be encoded more easily on separate lines in the input of the symbolic substitution system (since is is a correlator) than in the neural network system. The symbolic substitution system easily extends to accommodate a large number of facts since it is a correlator. Conversely, the neural network system is not shift invariant and if more facts are included by adding additional rows of neurons, the interconnection pattern required becomes much more complex (in general a 4-D matrix is required to implement all possible 2-D interconnections). When several sets of similar situations (each with the same sets of possible facts and rules) are to be processed, the symbolic substitution correlator approach has a clear preference (since its full correlation capability and shift invariance can now be utilized, specifically, we can search M parallel sets of input facts with the same set of rule filters), whereas the neural network requires a significant increase in the dimensionality and complexity of the required interconnection pattern. The symbolic substitution correlator also easily allows control over the output correlation peak intensities, whereas the neural network system requires multiple replicated inputs to easily allow uniform threshold values for the output neurons.

In terms of the number of facts that can be accommodated, near-term optical devices limit both systems to 10^6 facts (assuming 1000×1000 element binary input SLMs). This should be suitable for many applications, with combinations of multiple parallel architectures allowing extension to more facts. The storage capacity (number of rules that can be handled) is limited by the capacity of the volume hologram in the symbolic substitution system. This is expected to be less than the number of interconnections possible in the neural net system. Both architectures allow new rules to be inferred that are not explicitly encoded.

Space and frequency-multiplexed correlators have already been fabricated. Even though they are at a more mature level of development than are optical neural network architectures, it would be premature to select the symbolic substitution architecture over the neural network one for this application.

Thus, no definite conclusion is possible on the preferable choice of one system. This can only be addressed after a detailed analysis of the mask required in the neural network and the positioning requirements of the symbolic substitution correlator.

We gratefully acknowledge the Defense Advanced Research Projects Agency and NASA Ames for their support of this research.

References

1. C. Warde and J. Kottas, "Hybrid Optical Inference Machines: Architectural Considerations," Appl. Opt. **25**, 940 (1986).
2. H. J. Caulfield, "Optical Inference Machines," Opt. Commun. **55**, 259 (1985).
3. H. H. Szu and H. J. Caulfield, "Optical Expert Systems," Appl. Opt. **26**, 1943 (1987).
4. G. Eichmann and H. J. Caulfield, "Optical Learning (Inference) Machines," Appl. Opt. **24**, 2051 (1985).
5. A. D. McAulay, "Real-Time Optical Expert Systems," Appl. Opt. **26**, 1927 (1987).
6. J. A. Anderson, "Cognitive Capabilities of a Parallel System," in *Disordered Systems and Biological Organization*, E. Bienenstock *et al.*, Eds. (Springer-Verlag, Berlin, 1986), p. 209.
7. N. J. Nilsson, *Principles of Artifical Intelligence* (Springer-Verlag, Berlin, 1980).
8. J. Lowrie, M. Thomas, K. Gremban, and M. Turk, "The Autonomous Land Vehicle [ALV] Preliminary Road-Following Demonstration," Proc. Soc. Photo-Opt. Instrum. Eng. **579**, 336 (1985).
9. D. Casasent and E. Botha, "Knowledge in Optical Symbolic Pattern Recognition Processors," Opt. Eng. **26**, 34 (1987).
10. A. Huang, "Parallel Algorithms for Optical Digital Computers," in *Proceedings, Tenth International Optical Computing Conference*, IEEE Catalog No. 83CH1880-4 (1983), p. 13.
11. E. Botha, D. Casasent, and E. Barnard, "Optical Symbolic Substitution Using Multichannel Correlators," Appl. Opt. **27**, 817 (1988).
12. D. Casasent and E. Botha, "Multifunctional Optical Logic, Numerical and Pattern Recognition Processor," in *Proceedings, AIAA Computers in Aerospace VI Conference* (Oct. 1987), p. 213.

CHAPTER 10

"An Adaptive-Clustering Optical Neural Net"

Submitted to APPLIED OPTICS, August 1989

An Adaptive-Clustering Optical Neural Net

David Casasent and Etienne Barnard

Carnegie Mellon University
Center for Excellence in Optical Data Processing
Department of Electrical and Computer Engineering
Pittsburgh, PA 15213

Abstract

Pattern recognition techniques (for clustering and linear discriminant function selection) are combined with neural-net methods (that provide an automated method to combine linear discriminant functions into piecewise linear discriminant surfaces). The resulting "adaptive-clustering neural net" is suitable for optical implementation and has certain desirable properties in comparison with other neural nets. Simulation results are provided.

I. Introduction

Artificial neural networks have received much recent attention[1, 2, 3] and various optical realizations[4, 5] of the classic backpropagation neural network[6] have been suggested. Various other optical neural network architectures have been described[7, 8, 9] and some[10, 11, 12, 13] have been demonstrated conceptually. In this paper we distinguish between optimization and adaptive learning neural networks (Sect. II) and we discuss various neural-net issues as background. We then advance a new "adaptive-clustering neural network" (ACNN) in Sect. III. Simulation results (performed on a Hecht-Nielsen Corporation electronic neural network) are then presented (Sect. IV), optical realizations of the ACNN are discussed (Sect. V) and a summary is advanced (Sect. VI). This ACNN uses a new learning algorithm that combines standard pattern recognition techniques and neural-net concepts to arrive at a new and quite useful method for neural network synthesis that can be realized optically with attractive results and potential.

II. Artificial Neural Networks

We distinguish between two main classes of neural networks[14, 15]: optimization neural nets and adaptive learning neural nets. Optimization neural nets are well understood and their basic theory is well established[16, 17]. Associative processors are another class of neural networks[16, 18, 19, 20, 21] that are also well understood. In this paper we consider adaptive learning neural nets. The major advantage of a neural net in multiclass pattern recognition is its ability to compute nonlinear decision surfaces (typically combinations of linear decision surfaces) for complex multiclass decision problems. In fact, many neural-net classifiers can create decision boundaries of arbitrary shape. Our proposed neural net uses this feature of neural nets in conjunction with initial weights selected using class prototypes of clusters - hence we refer to this as an adaptive-clustering neural net. It employs a three-layered architecture, consisting of input, hidden and output layers with interconnections between the input and hidden layers, and between the hidden and output layers.

II.1. Neuron representation spaces and dimensionality

To maintain a reasonable number of input (P_1) neurons, we recommend[14, 15] that the neuron representation space be an appropriate feature space. For image recognition applications, the feature space should not be pixel-based. Other feature spaces have the additional advantage that they can be made invariant to transformations such as in-plane rotations. This greatly reduces the number of training images required (i.e. we need not train on transformed versions of the objects to be identified). For an M-dimensional feature space, we use $M+1$ input neurons. The additional neuron is used to incorporate the threshold of the hidden-layer neurons into the input vector with the state of this neuron set to unity. We now detail this. A linear discriminant function (LDF) in a feature space described by feature vectors $\mathbf{x}$ can be written as

$$g(\mathbf{x})=\mathbf{w}^t\mathbf{x}+w_0, \tag{1}$$

where $\mathbf{w}$ defines the orientation of the linear decision boundary and w_0 defines its offset or location. When decisions depend on whether $g \gtrless 0$, then $-w_0$ is the threshold for the vector-inner product (VIP) $\mathbf{w}^t\mathbf{x}$. By adding an additional "1" to the feature vector $\mathbf{x}$ to produce $\mathbf{y}$, we include w_0 in $\mathbf{w}$ and we can now write Eq. (1) as

$$g(\mathbf{x}) = \mathbf{w}^t\mathbf{y}. \tag{2}$$

The number of neurons in layer two (hidden layer) is generally chosen empirically. The number of hidden-layer neurons determines the complexity of the decision surface. Thus too few neurons lead to poor classification performance, since a decision surface of complexity sufficient to separate the various classes cannot be created. In most neural nets, the use of too many hidden neurons is wasteful of resources and leads to poor generalization. By this we mean that the decision surfaces are adapted to the peculiarities of the training set.

Local minima are a frequent topic of discussion associated with the number of hidden neurons used. A local minimum is a value of the energy function that is a minimum in a local region, rather than being a global minimum. In training a backpropagation (BP) neural net[6], the initial state of the hidden-layer neurons is random and a given error rate and some energy is obtained. When training is repeated with different initial hidden neuron states, if a different error rate results, a local minimum exists. One must vary the number of hidden neurons and retrain with different initial conditions to empirically determine the number of hidden neurons. The presence of such variables results in long training times for neural nets (as various numbers of layer-two neurons and various starting conditions are tried) and it can result in a neural net that cannot easily be generalized to test data.

Local minima occur when hidden neurons become redundant during training (e.g. two of

the N hidden neurons encode decision boundaries that lie very close to one another). If each neuron encoded a distinct decision boundary, a lower error rate would result (if the number of neurons were too few). When the number of distinct hidden neurons is sufficient (equal to or greater than the minimum required), there is no effect on *classification* performance, since sufficiently complex decision surfaces can be created despite redundancies in the hidden neurons. Thus, in this case local minima are not of concern. Many researchers have found that extensive methods to produce 100% classification on training data are not merited, since test set performance often does not reflect such improved training set results. Recent work[22, 23] on the choice of the number of hidden neurons has concentrated on the case when the training samples are in random positions in the feature space, which is almost never the case in real pattern-recognition problems.

Thus, although local minima are not of major concern, an alternate technique to determine the number of hidden neurons with significantly reduced effort is a significant concern. Our new neural net addresses this issue by an organized procedure that selects the number of hidden neurons based on the number of clusters present in the multiclass data to be separated (as detailed in Sect. III).

The number of neuron layers used is another variable. For BP, it has been shown[24, 25] that any decision su .ce can be approximated to arbitrary accuracy with a three-layer neural net. Four-layer neural nets can also produce any such decision surface, but they are harder to train (since the Hessian of the criterion function with respect to the weights is more ill-conditioned when more layers are used) and they generally introduce more parameters that must be empirically selected. Since our neural net also approximates any such decision boundary with three layers, we restrict attention to a three-layer neural net.

The number of output-layer neurons equals the number of classes.

II.2. Criterion or error functions

One of the most popular adaptive learning neural nets is backpropagation (BP)[6]. The problems with this neural net are that it requires a large training set and long training time, and does not necessarily converge to the best minimum. Backpropagation is an example of a neural net which is trained by the minimization of an error or criterion function. The form of the error function that is minimized for such nets can affect performance and training time (e.g., the error function with the best error rate is often the one for which it is most difficult to reach a minimum error[26]). Standard BP uses an error function based on a sigmoid transfer function, while our ACNN uses the perceptron error function in training. We recently provided[26] a comparison of various error or criterion functions. It was shown that, in general, the use of a perceptron criterion function provides faster convergence with comparable error rates P_e to those obtained with the more popular sigmoid criterion function. The error function choice is not of major concern in the performance of BP and our ACNN (it is included to note the differences between BP and ACNN and because the criterion function used specifies the type of linear classifier employed, as we detail in Sect. III).

II.3. Update algorithm

One reason for the slow convergence of BP is that a gradient-descent (delta rule) algorithm is often used to update the weights in training. Our ACNN uses a conjugate-gradient algorithm[27] for weight update since it is faster and does not require the empirical choice of parameters such as the learning rate and momentum[26, 28]. In conjugate-gradient updating, all of the training set data are fed to the system (once) and then the weights are updated. Conversely, with gradient descent the weights can be updated after the presentation of each sample in the training set. A batch type of gradient-descent algorithm can also be used, with weights updated only after all training data have been presented to the system once. Generally, batch gradient

descent has the slowest convergence (since the parameters cannot be updated and selected at different steps). Sequential (non-batch) gradient descent generally performs better than batch gradient descent, since it makes more steps toward the solution (in one presentation of the training set of data). However, selection of its parameters is empirical and we have found that conjugate-gradient optimization performs better. We attribute this to the fact that conjugate-gradient optimization adapts the learning parameters in a sensible way, whereas these parameters are kept fixed or adapted heuristically for gradient descent.

In difficult multiclass decision problems we have found conjugate-gradient training to be much more efficient than gradient descent. With neural net hardware and software (such as the Hecht-Nielsen Corporation AZP which we use) conjugate-gradient optimization is very attractive. In our comparisons of BP and the ACNN we use the same conjugate-gradient algorithm to update the weights.

II.4. Initial weights

Another reason for the long training time for BP is that the initial weights are chosen arbitrarily. In our ACNN algorithm, the initial weights are set using pattern-recognition techniques and then they are refined using neural-network techniques. This is a major reason for the improved performance of our ACNN. We have tested BP using initial weights chosen from clustering techniques similar to those used for the initial weights of the ACNN. We found[29] negligible improvement in training time and worse performance in some cases. We attribute this to the fact that BP can sometimes use hidden neurons in more sophisticated ways than is the case in the hidden layer of our ACNN and that this cannot be achieved when a preset weight choice is used.

This present section was intended to highlight issues associated with neural networks and

to note differences between our algorithm and the more extensively tested and analyzed BP algorithm.

III. Adaptive clustering neural net (ACNN) training algorithm

Our three-layer ACNN is shown in Fig. 1. It is similar to the standard multilayer perceptron. We now detail its design and use for multiclass pattern recognition. The input (P_1) neurons are analog and represent a feature space which can be of low dimensionality (we add an additional feature which is always kept at unity to adapt the threshold of the hidden neurons as well). The hidden-layer neurons at P_2 correspond to clusters in feature space, with several clusters (neurons) used for each class in a multiclass application. The P_1-P_2 weights are used to assign an input to a cluster. We typically use two to five clusters per class. The layer two neurons are binary and (in testing) the P_2 neuron with the largest input activity fires and denotes the cluster to which the input belongs. During training the P_1-P_2 weights adapt as we will detail (we employ a conjugate-gradient algorithm) and thus refine our initial weight estimates. The hidden layer to output weights are fixed (all are either zero or one) and perform the mapping of the P_2 clusters to one of the classes (with one P_3 neuron assigned per class of data). Thus, we initially assign several layer-two "cluster neurons" to each class and use fixed P_2-P_3 weights to assign each P_2 cluster to a final class (output neuron in P_3). This is attractive and new since it allows us to use standard clustering and pattern-recognition techniques to select the initial P_1-P_2 weights (initial LDFs) and new neural-net techniques to adapt or refine these weights. We employ a perceptron criterion or error function (this defines our LDFs) rather than a sigmoid error function, since faster convergence with a comparable error rate is obtained.

There are no commonly-used standard (non-neural net) techniques to obtain piecewise linear decision surfaces for two- or multiclass problems (except nearest-neighbor methods). Because of the importance of neural-net techniques in addressing this problem, and since we use

nearest-neighbor techniques in selecting our clusters, we briefly review standard multiclass techniques. In a nearest-neighbor classifier, the distance between an input and all training samples is calculated and the input is assigned to the class of the closest training sample. From tests on all training data in each class, the bounds on each class are determined and one can obtain piecewise linear decision surfaces. However, the nearest-neighbor technique is computationally intensive (requiring calculation of the distance to all training samples). Conversely, neural nets have a long training time (which is off-line and of less concern) but their classification times (an on-line requirement) are short. In addition, all training samples must be stored for a nearest-neighbor system and thus storage requirements can be excessive. Finally, nearest-neighbor systems do not perform well when the probability-density functions of the classes overlap significantly. The calculation of the K nearest neighbors is useful here (the input is assigned to the class to which the majority of these K samples belong). However, the selection of K is empirical.

Two other multiclass techniques are Gaussian and linear classifiers. Gaussian classifiers assume that the data in each class are normally distributed and for each class its mean and variance are estimated. To classify an input vector, a posteriori probabilities are calculated for each class with Bayes' rule, and the input is assigned to the class with the highest probability. This technique (and all parametric methods) work only if the data follow the assumed distribution and this is rarely the case. To produce multiclass decision boundaries with LDFs, the mean vector $\mathbf{m}_c$ of each class can be calculated and used as an LDF. The VIP of the input with each $\mathbf{m}_c$ and thresholding denotes the class estimate for the input. Criterion functions (error functions) represent a preferable way to select an LDF for each class. One can employ pairwise LDFs (for each LDF, some class i is compared with another class j). These approaches are computationally intensive and not attractive for problems with many classes and they may lead to decision surfaces that have undefined regions (not corresponding to any class).

Thus, standard linear-discriminant techniques for multivariate pattern recognition allow us to determine suitable linear discriminants, but these are generally not powerful enough for realistic pattern-recognition applications that require nonlinear decision surfaces. In our ACNN, neural-net techniques provide refinements to the linear-discriminant weight estimates and automatically combine many linear decision boundaries into piecewise linear decision boundaries. We now detail the design and update rules for our ACNN.

III.1. Selection of the number of hidden layer (cluster) neurons

To select the prototypes/exemplars or cluster representatives we use two steps. As our prototypes we desire the N prototypes in the training set whose removal cause the most error in a nearest-neighbor classification. We assume a large training set (N_T samples) for our multiclass problem (so large that simple clustering techniques cannot produce a suitable set of clusters). We first use standard techniques[30] for sample-number reduction to obtain a modest number of prototypes N_R. This "reduced nearest-neighbor" clustering technique divides the N_T samples into two groups (A and B), where the samples in A classify all N_T samples correctly using a nearest-neighbor technique. Initially, all samples are in group B. The samples in A are used as the prototypes in a nearest-neighbor classifier. Each sample in B is sequentially presented to the nearest-neighbor classifier. If it is incorrectly classified, it is added to A. This procedure is repeated until the samples in group A can correctly classify all N_T samples. (Typically around 5% to 30% of the training samples are still present in N_R and this is still too large a number of P_2 neurons.)

Thus we employ a second step to further reduce the number of prototypes (clusters) to an acceptable number N. To achieve this we remove the first prototype, use the remaining N_R-1 samples in a nearest-neighbor classifier to classify the N_T original samples, and calculate the number of misclassifications. We then remove only the second prototype, and repeat the above

procedure with the remaining N_R-1 samples. This procedure continues until the removal (separately) of each of the N_R prototypes has been tested. If N is prespecified, we keep the N prototypes whose removal would cause the most errors. We can also use the number of errors obtained by removing each prototype to select N (i.e. we select N that results in no more than a given error rate or for which there is a jump in the number of errors produced). We insure that at least one prototype is chosen from each class. Insuring that we keep one prototype per class has not been a problem in our benchmarks (i.e. if the prototypes are ordered by their error rate, we do not find a number of consecutive prototypes in one class before one from another class occurs). In our initial benchmarks we have not found significant branch points or jumps in the error rates of the ordered samples. There is also no restriction that the same number of prototypes be selected from each class (the data will determine this). Considerable flexibility is possible in how the N prototypes are selected since training will refine the initial choices, and thus this issue is not of major concern.

This procedure does not account for the fact that, when several samples are not included as prototypes, performance will be worse than when only one of the samples is omitted. However, the purpose of selecting prototypes (or cluster representatives) is only to provide a reasonable or approximate initial selection (the neural net adaptations of these initial choices address the global problem).

We note that use of a nearest-neighbor technique for training is acceptable, but it is not suitable for classification (where on-line real-time requirements exist). The combination of our nearest-neighbor prototype selection and ACNN update algorithm will be shown to require fewer iterations than BP. To quantify the significance of this, we now briefly address the number of operations required to select prototypes and relate it to the number of operations required in one BP iteration on all N_T training samples. For each sample, our prototype selection algorithm

must calculate the distance to all other points in the training set. For all N_T samples, the calculation of the distances from all points to all points (i.e. the number of distance calculations required for one pass through the N_T training samples) is approximately $0.5{N_T}^2$ (we precalculate this once and use the 0.5 factor since the calculations are symmetric). In BP, all N_T samples are presented and after each sample we must calculate the activities of all N neurons (N hidden neurons are assumed and the calculation of the activities of the output neurons is ignored), i.e. $N_T N$ calculations are required. The calculation times for the operations in the two cases are equivalent, each is a VIP of dimension equal to that of the feature space used (the calculation times for each operation are exact for the case of layer 1 and 2 neurons). If the additional number of BP iterations required is I, then for our algorithm to be computationally efficient, we require

$$0.5{N_T}^2 < N_T N I. \tag{3}$$

Since $N_T \gg N$ our algorithm may not offer a significant advantage in training time (once N is fixed in BP) unless I is very large.

In obtaining the result in Eq. (3), we assumed that all N_T samples were used in selecting the N prototypes. We have found that we need only use approximately $5N$ randomly selected samples from the full set of N_T in our prototype selection (N is the number of prototypes or cluster neuron used at P_2 and we have always found that 2 to 5 neurons per class suffice). Thus, we employ our algorithm using $5N$ samples (not N_T). The inequality to be satisfied is now

$$\begin{aligned} 0.5(5N)^2 &< N_T N I \\ 25N &< 2N_T I. \end{aligned} \tag{4}$$

To further evaluate this, we assume $N_T \approx 100N$ (this is quite typical for distortion-invariant problems to adequately represent all distortions). We then find

$$25 < 200I \tag{5}$$

which is independent of N. This inequality is always satisfied. As we shall see, BP has always required at least on the order of I=100 more iterations of the full training set than has our ACNN algorithm. In this case

$$25 < 2\times10^4, \tag{6}$$

and the computational time savings of our algorithm is quite significant.

Thus, to summarize, in the two steps of our prototype selection algorithm we use $5N$ random samples from the full N_T set. We select the number of hidden neurons N to be 2 to 5 times the number of classes (depending on the difficulty of the problem). Section IV details these choices for two examples.

III.2. Initial P_1-P_2 weights

We now address how we select the initial P_1-P_2 (input-to-hidden layer) weights. We denote the weight between input neuron j and hidden neuron i by w_{ij}. We denote the vector position of prototype i in our D-dimensional feature space by $\mathbf{p}_i$ (i.e. this is the feature vector for prototype i) and element j of it by p_{ij}. We can now describe the input weights from P_1 to P_2 as

$$w_{ij} = \begin{cases} p_{ij} \text{ for } j=1,\ldots,D \\ -(1/2)\sum_{i=1}^{D} p_{ij}^2 \text{ for } j=D+1. \end{cases} \tag{7}$$

The first D (out of D+1) elements of each weight vector from P_1 to layer-two neuron i are thus the feature vector $\mathbf{p}_i$ associated with that prototype. The last (D+1) input neuron activity is always "1" and its weight to hidden layer neuron i is associated with its LDF threshold. We choose these initial weights since they ensure that the classifier initially implements a nearest-neighbor classifier based on the prototypes, as we now detail.

Each hidden neuron i has connections from all D+1 input neurons and thus has a weight vector $\mathbf{w}_i$ associated with it. For an input $\mathbf{x}_a$, the input to neuron i in layer two is

$$\mathbf{w}_i{}^t\mathbf{x}_a = \mathbf{p}_i{}^t\mathbf{x}_a - 0.5\, p_{ij}{}^2 \tag{8}$$

where the first term is the contribution to the VIP from the first D weights and the last term is the contribution due to the additional D+1 input neuron. We rewrite Eq. (8) as

$$\begin{aligned}\mathbf{w}_i{}^t\mathbf{x}_a &= (0.5)2\mathbf{p}_i{}^t\mathbf{x}_a - 0.5\mathbf{p}_i{}^t\mathbf{p}_i + 0.5\mathbf{x}_a{}^t\mathbf{x}_a - 0.5\mathbf{x}_a{}^t\mathbf{x}_a \\ &= 0.5[\mathbf{x}_a{}^t\mathbf{x}_a - (\mathbf{p}_i{}^t\mathbf{p}_i - 2\mathbf{p}_i{}^t\mathbf{x}_a + \mathbf{x}_a{}^t\mathbf{x}_a)] \\ &= 0.5(\|\mathbf{x}_a\|^2 - \|\mathbf{p}_i - \mathbf{x}_a\|^2).\end{aligned} \tag{9}$$

From Eq. (9) we see that the VIP is related to the Euclidean distance (denoted by $\|\ \|$) between the input $\mathbf{x}_a$ and the prototype $\mathbf{p}_i$ associated with hidden neuron i. The choice of weights in Eq. (7) thus achieves nearest-neighbor classification since it ensures, from Eq. (9), that the hidden neuron closest to $\mathbf{x}_a$ will have the largest input (since the second term in Eq. (9) is then smallest) and will be most active.

III.3. Training (weight update) algorithm

We now detail how we update the initial P_1-P_2 weights to achieve improved piecewise linear decision surfaces. We input each of the full N_T set of training vectors $\mathbf{x}_a$. For each $\mathbf{x}_a$ we calculate the most active hidden neuron $i(c)$ in the proper class c and the most active one $i(\bar{c})$ in any other class ($\bar{c}$). We denote the weight vectors for these two layer-two neurons by $\mathbf{w}_{i(c)}$ and $\mathbf{w}_{i(\bar{c})}$ and their VIPs with the input by $\mathbf{w}_{i(c)}{}^t\mathbf{x}_a$ and $\mathbf{w}_{i(\bar{c})}{}^t\mathbf{x}_a$. The perceptron error function (criterion function) E_P used is shown in Fig. 2. The solid (dashed) curves correspond to the true (false) classes 1 and 2 cases. The offset S is a safety margin that forces training set vectors which are classified correctly by a small amount (less than S) to also contribute to the criterion function. As discussed elsewhere[26] we chose S=0.05 (all features were normalized between 0 and 1). The use of S forces the classifier to try to classify all training samples correctly by at least an

amount S, improving test set performance (and thus generalization).

For each training sample in N_T we add an error (penalty) to E_P The error added is

$$E = 0 \text{ if } \mathbf{w}_{i(c)}{}^t\mathbf{x}_a > \mathbf{w}_{i(\bar{c})}{}^t\mathbf{x}_a + S \qquad (10)$$
$$S + (\mathbf{w}_{i(\bar{c})} - \mathbf{w}_{i(c)})^t\mathbf{x}_a \text{ otherwise},$$

where the E=0 case corresponds to the situation when the proper layer-two neuron is most active (by an amount S above the most active false neuron) and where the other case corresponds to the situation when the false class VIP is larger than the true class VIP, or within S of it.

After all N_T training samples have been run through the system, we accumulate all of these errors or energies (all are positive or zero). We also accumulate the gradients $\nabla_{\mathbf{w}_i} E$. From Eq. (10), by taking the derivative with respect to $\mathbf{w}_i$, we see that $\nabla_{\mathbf{w}_i} E$ is zero for all i when an input is classified correctly by more than S; otherwise, it equals either $\mathbf{x}_a$ (if input a should be classified into the same class as cluster-neuron i) or $-\mathbf{x}_a$ (if a is incorrectly classified by cluster neuron i). Thus, the sum of all the contributions to $\nabla_{\mathbf{w}_i} E$ equals the sum of the $\pm\mathbf{x}_a$ for samples erroneously classified (or correctly classified but with a margin less than S) in layer-two clusters. We then use $\nabla_{\mathbf{w}_i} E$ to adapt the weights $\mathbf{w}$ by the conjugate-gradient algorithm. We then repeat presentation of the training set (a new iteration), calculate the new errors E and $\nabla_{\mathbf{w}_i} E$ and update the weights accordingly. This procedure repeats until satisfactory performance on the test set is obtained.

We considered other LDFs (Ho-Kashyap, Fisher, Fukanaga-Koontz etc.). However, these LDFs require more calculation than our current algorithm does to update the weights. Thus, for computational reasons, our present choice (perceptron criterion) is preferable.

III.4. Input P_1 neuron representation space

In our distortion-invariant multiclass pattern recognition applications, we use a wedge-sampled magnitude Fourier transform feature space[31], since this feature space can easily be produced optically. Fig. 3(a) shows the standard architecture that produces the Fourier transform at P_2 of the P_1 input 2-D image data. Fig. 3(b) shows the standard wedge-ring detector used at P_2. The wedge features provide scale invariance and the ring features provide in-plane rotation invariance. Our distortion-invariant data will involve different aspect views of several objects (and not in-plane distortions). Thus we chose the wedge features (this provides scale invariance, although we do not include scale distortions in our test data). We obtain aspect-view invariance by training on various aspect-distorted object views.

IV. Test results

We consider two databases: an artificial set of data[15, 26] (to demonstrate the nonlinear surfaces produced using only two features) and a set of three aircraft with various azimuth and elevation (3-D) distortions present. We refer to these as benchmarks 1 and 2.

IV.1. Benchmark 1 results (artificial data)

An artificial set of 383 samples in three classes (181 in class 1, 97 in class 2 and 105 in class 3) with 2 features was generated with samples as shown in Fig. 4. This problem definitely requires a nonlinear decision boundary and the results can be shown in the 2-D feature space. This is the purpose of this example, since no separate test data exist. The neural net used contained three input neurons (two for the features plus one for the threshold), six hidden neurons (two per class) and three output neurons (one per class). All N_T samples were used to select the prototypes. The first "reduced nearest-neighbor clustering" produced 31 prototypes (8.1% of the total N_T) that gave an error rate P_e=0% for all samples. The six prototypes whose removal gave the most error were then selected in stage two.

After 80 iterations of the full training set, the classification rate (defined as the percentage of test samples correctly classified) was constant at 97.1% with our ACNN algorithm . After 300 iterations the BP classification rate was constant at approximately the same value (96.3%). (This result is the average obtained over 10 runs with different random initial weight sets.) The final input layer weights to the six hidden layer neurons correspond to six straight lines (LDFs) in the feature space. For BP these six lines would define the decision surface. In the ACNN this is not the case (because of the winner-takes-all action at P_2). The decision-surface lines were determined by successively providing all of the possible feature vectors on a grid of x_1–x_2 values (for both x_1 and x_2 in the interval [0,1]) to the classifier, and for each feature vector determining the class into which it is classified by the neural net. The decision boundaries indicate where a transition in classification occurred. The boundaries thus obtained are shown in Fig. 5. They produce four separate regions of feature space (two correspond to the same class and the others correspond to the other two classes).

From inspection of Fig. 4 one would estimate that a piecewise-linear decision surface with at least five straight-line sections would be needed to separate the data adequately and that about ten errors might be expected. Thus at least five hidden neurons are expected to be needed. In Fig. 4 we see that, with six hidden neurons, approximately 10 classification errors are made, producing the error rate of 97.1%.

Figure 6 compares the classification rate for the two neural nets and for a multivariate Gaussian classifier. Both neural nets give comparable classification rates (97.1% and 96.3%) after convergence, whereas the Gaussian classifier's performance is worse (89.5%) and by definition does not vary with the number of iterations of the training set. The speed of learning of the ACNN is much faster (convergence in 80 iterations) than for BP (approximate convergence in 300 iterations). From Eq. (4) this represents approximately an additional

$N_T NI$=(383)(6)(220)=505560 VIP calculations required with BP. The prototype selection steps in our ACNN algorithm required approximately $0.5{N_T}^2$=(0.5)$(383)^2$≈73350 VIPs and thus the total number of calculations and hence training time for our ACNN is considerably less than the learning time for BP. We reran the prototype selection portion of our ACNN algorithm using only $5N$=30 samples randomly selected from the 383, insuring that we obtain at least one prototype per class. The decision boundaries produced are shown in Fig. 7. As can be seen, the decision boundaries are virtually identical; the resulting error rates differ by only 0.2% (96.9% classification was obtained after 100 iterations). This was now achieved with only $0.5(30)^2$=450 VIPs for prototype selection.

This data set therefore indicates that similar performances can be obtained with BP and ACNN, with ACNN training appreciably faster than BP. We have also seen that the time for prototype selection with ACNN can be made negligible by using a reduced number of learning samples, without affecting performance adversely.

IV.2. Benchmark 2 results (3-D distorted aircraft data)

As our second data set, we used synthetic distorted aircraft imagery and our wedge-sampled Fourier feature space. The imagery used were three aircraft (F-4, F-104 and DC-10) binarized to 128x128 pixels with each aircraft occupying about the central 100x64 pixels. As our training set, we used 630 images of each aircraft (a total of N_T=1890 training set samples). The images were different azimuth views (with the aircraft viewed from different angles left to right) and elevation views (with the aircraft viewed from different angles above or below its center line). The range of azimuth angles used covered -85° to +85° and the elevation angle was varied from 0° to 90° with 5° increments in each angle (the same image results if negative elevation angles are used). The input neuron representation space was a 32 element feature space (the 32 wedge magnitude Fourier samples). The test set used consisted of 578 orientations of

each aircraft not present in the training set (these were views at internal angles about 2.5° in each direction from those in the training set). Fig. 8 shows three distorted versions of each aircraft. The left image is the top-down view with 0° variation in elevation and azimuth. The central image shows a view from an azimuth angle of 45° to the left. The right image for each object shows an image with elevation angle of 45°.

The three-layer ACNN used contained 33 input neurons, 9 hidden neurons and three output neurons (one per class).

Fig. 9 compares the speed (number of iterations of the full training set) and classification performance for the two neural nets and the Gaussian classifier. Both neural nets yield the same classification rate (98.6%) compared to only 89% for the Gaussian classifier. BP converges in 350 iterations and our ACNN in fewer (180) iterations. As with the two-dimensional data set, a reduced data set for prototype selection can be employed successfully. It was found that with $5N$=45 samples used for prototype selection, 98.6% classification performance was obtained after 180 iterations. With this reduced number of samples, the time for prototype selection is negligible compared with the time for a single iteration, so that the relative training times are again determined by the number of iterations required for each method. Thus, ACNN requires approximately 50% of the training time of BP.

V. Optical and optical/electronic realization

Many choices are possible for the role of optics in the learning and classification stages of our ACNN. These are now discussed. The feature space (wedge-sampled magnitude Fourier transform) should be optically calculated (even in learning) since this feature space is easily produced optically[32, 33] and since we will use the optically produced feature space in our on-line classification. The two steps of prototype selection are best performed electronically - since they

are off-line operations and require manipulation of stored data and control operations most compatible with digital electronics. The distance calculations required in the nearest-neighbor calculations can be performed on an optical VIP architecture (we now discuss this and the use of optics in the learning stage).

Once the initial P_1-P_2 weights have been chosen, the learning stage can be implemented in optics or electronics. Fig. 10 shows one such architecture. The input sample $\mathbf{x}_a$ is entered at P_1 (on LEDs, laser diodes or a 1-D spatial light modulator (SLM)). It is imaged onto the initial set of N weight vectors (for the N prototype hidden layer neurons) which are arranged on rows at P_A (with the first two to five rows corresponding to the prototypes for class 1, the next two to five rows being the prototypes for class 2, etc.). Thus the rows at P_A are the initial weights as in Eq. (7). The VIPs of $\mathbf{x}_a$ and all of the $\mathbf{w}_i$ weight vectors at P_A are formed on a linear detector array at P_2. The P_A rows and P_2 elements are separated into C groups (the C classes). The maximum VIP element in each class is determined (simple comparator logic is sufficient since the number of prototypes per class is small). This provides us with $\mathbf{w}_{i(c)}{}^t\mathbf{x}_a$ and $\mathbf{w}_{i(\bar{c})}{}^t\mathbf{x}_a$ in Eq. (10). Bipolar values for $\mathbf{w}_i$ should be handled by spatial multiplexing at P_A and subtraction of adjacent P_2 outputs. Alternatively, the P_A data can be placed on a bias (but this increases dynamic-range requirements). The weights must be updated after each iteration of the training set. If P_A is a microchannel spatial light modulator[34] (or similar device) that can record positive and negative data (with a bias on the device), we can update the weights by adding and/or subtracting the appropriate values for each weight. These updates to the weights at P_A are various combinations of the training vectors $\mathbf{x}_a$. These could be calculated in electronics, entered sequentially at P_1 and (with a mechanism to activate only selected rows at P_A) we could update P_A as required. Alternatively, we could repeat each $\mathbf{x}_a$ at P_1 and vary the input illumination and the P_A row accessed and hence control the amount of each $\mathbf{x}_a$ added to or subtracted from each weight vector at P_A. The digital control required, the complexity of the system (a modulated

light source to control the amount of each $\mathbf{x}_a$ used, access to only one row of P_A at a time), the need for N accesses of P_A for each of the N_T vectors $\mathbf{x}_a$, and the P_A SLM requirements make the electronic calculation of the updated weight vectors and the electronic off-line implementation of the learning stage preferable (at present). As P_A SLM technology matures, it would probably be realistic to calculate all VIPs optically, determine the new weights electronically, and reload these directly into P_A after each iteration of the training set. However, at present, we assume that all learning is electronic (since it is off-line).

Once learning has been completed, the P_1-to-P_2 weights are fixed and the input-to-hidden layer neurons and weights (the P_1-to-P_2 neuron system) can be implemented on an optical VIP system (such as P_1 to P_2 of Fig. 10) with a fixed mask at P_A. The number of P_1 neurons is modest (the input neuron representation is a compact feature space), and the number of P_2 neurons is also small (typically less than five times the number of classes). Our ACNN requires a winner-takes-all (WTA) maximum selection of the most active P_2 neuron. This can be implemented with a WTA neural network or in standard comparison techniques. Since the number of P_2 neurons (N) is small, standard electronic WTA techniques are preferable (we quantify this below). Since the P_2-to-P_3 hidden-to-output neuron weights are fixed and are all unity or zero, the P_2-to-P_3 weights simply perform a mapping and can easily be implemented in electronics. Thus, we implement the input-to-hidden layer neuron weights and calculations optically, and the hidden layer neuron maximum selection (WTA) and the hidden-to-output neuron mapping in electronics. Fig. 11 summarizes the learning and classification stages in block diagram form with attention to which operations are performed in optics and which in electronics.

The two WTA electronic techniques possible (in classification) are to use an operational amplifier peak detector to scan all N outputs at P_2 or to employ a parallel digital technique. In

the digital technique, the N outputs are A/D converted, each pair of P_2 outputs (1 and 2, 3 and 4, etc.) are pairwise compared and the maximum of each pair is obtained. Pairwise comparisons of the $N/2$ outputs are then performed and the procedure is continued for $\log_2 N$ levels until the maximum is obtained. For 100 input and hidden neurons, one matrix-vector multiplication (required to update the P_2 neuron activities) requires about 10,000 additions and 10,000 multiplications, whereas maximum selection requires only about 100 comparisons. Thus, the maximum-selection is typically negligible computationally compared with the neuron-update stage, and can be implemented in serial electronic hardware without sacrificing the speed of the system. We thus implement the WTA operation in electronics using comparators rather than with a neural net. The specific electronic WTA technique chosen depends on the accuracy and speed required. Since these operations are required once for each test input in classification, the WTA time required is set by the rate at which new input image data occurs and the rate at which its features can be calculated.

VI. Summary, conclusions and discussion

A new three-layer adaptive-clustering neural net (ACNN) has been described. It provides for a new procedure to select the number of hidden layer neurons (we use several neurons per class, each being a prototype or cluster representative of a particular class) and provides initial (non-random) input-to-hidden layer neuron weights. These initial weights are selected using standard pattern recognition clustering techniques. They are then updated during learning using a new neural net adaptive supervised learning algorithm. This results in a new neural net that combines standard pattern recognition and neural-net techniques to produce piecewise-linear decision surfaces from linear discriminant functions. The input neurons are analog and of low dimensionality (a feature space with inherent distortion invariances). Quantitative data show that the learning time and number of calculations required in our new ACNN is significantly faster (by a factor of 2 to 4) than the more well-studied BP neural net. We also found that the

use of a conjugate-gradient (rather than gradient descent) update algorithm significantly speeds up BP.

BP and the ACNN will usually not result in similar weights since BP uses neurons for other operations besides clustering, because BP has no WTA competition in its hidden layer as in the ACNN and because the hidden-to-output weights are different in BP and only perform mapping in the ACNN. However, the decision boundaries that result are usually very similar (with the ACNN decision boundaries generally being a piecewise-linear approximation to the more curved ones in BP). Thus the two classifiers employ different means to similar ends, with the ACNN providing faster training without the need to select many empirical parameters. Since only one hidden neuron in ACNN is dominant, piecewise-linear surfaces result and more hidden neurons may be needed. Our intent is not to compare BP and our ACNN, rather we note the attractive properties of our new neural net. Besides providing a new way to select the hidden neurons, our neural-net algorithm has only one ad-hoc parameter to be empirically selected (the number of hidden neurons). Changes in ACNN weights during training provide information on the data that can be of use in better understanding results and in extending results to other cases (other neural nets do not have this property). For example, in sequential gradient descent updating algorithms (the delta rule) different results occur depending on the order in which the training data are presented and depending on the random initial weights (by comparison, the ACNN provides consistent results).

Acknowledgment: This work was supported by a contract (DAAH01-89-C-04180) from the Defense Advanced Research Project Agency, monitored by the U.S.Army Missile Command.

List of figure titles

Figure 1: Adaptive-clustering neural net

Figure 2: Perceptron criterion function. S denotes the safety margin and the solid and dashed curves correspond to classes 1 and 2 respectively

Figure 3: Input P_1 neuron representation space (wedge sampled Fourier transform): (a) architecture (b) P_2 sampling

Figure 4: Three-class, two-feature artificial database example (Benchmark 1)

Figure 5: Nonlinear decision boundaries produced for the artificial database

Figure 6: Comparative data on speed of convergence for Benchmark-1 data

Figure 7: Nonlinear decision boundaries produced for the artificial database when prototypes are selected from reduced training set

Figure 8: Representative images for the three-class 3-D distortion example (Benchmark 2)

Figure 9: Comparative data on speed of convergence for Benchmark-2 data

Figure 10: Possible optical architecture for adaptive learning

Figure 11: Block diagram for adaptive-clustering neural net using (a) electronics for learning (training) and (b) optics for classification (on-line)

References

1. *International Conference on Neural Networks,* IEEE (Cat. No. 88CH2632-8), IEEE Press, San Diego, 1988.

2. *International Joint Conference on Neural Networks,* IEEE (Cat. No. 89CH2765-6)/International Neural Network Society, IEEE Press, Washington, DC, 1989.

3. *Advances in neural information processing systems,* M.Kaufmann, Boulder, Colo., 1988.

4. K.Wagner and D.Psaltis, "Multilayer optical learning networks", Applied Optics **26**, pp. 5061-5076 (1987).

5. H.Yoshinaga,K-I.Kitayama and T.Hara, "All-optical error-signal generation for BP learning in optical multilayer networks", Opt. Lett. **14**, pp. 202-204 (1989).

6 D. E. Rumelhart,G. E. Hinton and R. J. Williams, "Learning internal representations by error propagation, " in *Parallel Distributed Processing.* Cambridge MA: MIT press, pp. 318-362,ch. 8,1986.

7. Nabil Farhat, "Optoelectronic Analogs of Self-Programming Neural Nets: Architecture and Methodologies for Implementing Fast Stochastic Learning by Simulated Annealing", Applied Optics **26**, pp. 5093-5103 (1987).

8. D.Z.Anderson and D.M.Lininger, "Dynamic optical interconnects: volume holograms as two-port operators", Applied Optics **26**, pp. 5031-5038 (1987).

9. A.D.Fisher, W.L.Lippincott and J.N.Lee, "Optical implementations of associative networks with versatile adaptive learning capabilities", Applied Optics **26**, pp. 5039-5054 (1987).

10. D. Psaltis and N. Farhat, "Optical information processing based on an associative-memory model of neural nets with thresholding and feedback", Optics Letters **10**, pp. 98-100 (1985).

11. B.H. Soffer, G.J. Dunning, Y. Owechko, and E. Marom, "Associative holographic memory with feedback using phase-conjugate mirrors", Optics Letters **11**, pp. 118-20 (1986).

12. L-S.Lee, H.M.Stoll and M.C.Tackitt, "Continuous-time optical neural network associative memory", Opt. Lett. **14**, pp. 162-164 (1989).

13. K.Hsu and D.Psaltis, "Invariance and discrimination properties of the optical associative loop", *International Conference on Neural Networks*, IEEE, San Diego, July 1988, pp. II-395 - II-402.

14. E.C.Botha, D.Casasent and E.Barnard, "Optical neural networks for image analysis: Imaging spectroscopy and production systems", *International Conference on Neural Networks*, IEEE, San Diego, July 1988, pp. I-541 - I-546.

15. E.Barnard and D.Casasent, "Image processing for image understanding with neural nets", *International Joint Conference on Neural Networks*, IEEE, Washington, DC, June 1989, pp. I-111 - I-116.

16. J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proc. Natl. Acad. Sci. USA **79**, pp. 2554-58 (1982).

17. J.J.Hopfield and D.W.Tank, "Computing with neural circuits: A model", Science **233**, pp. 625-633 (1986).

18. G.E. Hinton and J.A. Anderson, editors, *Parallel Models of Associative Memory*, Laurence Erlbaum Associates, Hillsdale, NJ, 1981.

19. B. Montgomery and B.V.K. Vijaya Kumar, "An Evaluation of the Use of the Hopfield Neural Network Model as a Nearest-Neighbor Algorithm", Applied Optics **25**, pp. 3759-66 (1986).

20. T.Kohonen,G.Barna and R.Chrisley, "Statistical pattern recognition with neural nets: benchmarking studies", *International Conference on Neural Networks*, IEEE, San Diego, July 1988, pp. I-61 - I-68.

21. B. Kosko, "Bidirectional Associative Memories", IEEE Trans. Syst., Man, Cybern. **18**, pp. 49-60 (1988).

22. E.B.Baum, "On the capabilities of multilayer perceptrons", J. Complexity **4**, pp.

193-215 (1988).

23. G.Mirchandani and W.Cao, "On hidden nodes for neural nets", IEEE Trans. Circuits and Systems **36**, pp. 661-664 (1989).

24. B.Irie and S.Miyake, "Capabilities of three-layered perceptrons", *International Conference on Neural Networks*, IEEE, San Diego, July 1988, pp. I-641 - I-648.

25. M.Stinchcombe and H.White, "Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions", *International Conference on Neural Networks*, IEEE, Washington, DC, June 1989, pp. I-613 - I-618.

26. E.Barnard and D.Casasent, "A comparison between criterion functions for linear classifiers, with an application to neural nets", IEEE Trans. Syst., Man, Cybern. **SMC-19** (1989).

27. M.J.D.Powell, "Restart procedures for the conjugate gradient method", Mathematical Programming **12**, pp. 241-254 (1977).

28. A.Lapedes and R.Farber, "How neural nets work", *Neural Information Processing Systems*, D.Z.Anderson,ed., AIP, Denver, Co, 1988, pp. 442-456.

29. E.Barnard,E.Botha and D.Casasent, "Neural nets as piecewise linear classifiers: new algorithms", Neural Networks **1**, pp. Supplement 73 (1988).

30. T.M.Cover and P.E.Hart, "Nearest neighbor pattern classification", IEEE Trans. Info. Theory **IT-13**, pp. 21-27 (1967).

31. G.G.Lendaris and G.L.Stanley, "Diffraction-pattern sampling for automatic pattern recognition", Proc. IEEE **58**, pp. 198-205 (1979).

32. H.Kasdan and D.Mead, "Out of the laboratory and into the factory - optical computing comes of age", *Proc. Electro Optical System Design*, 1975, pp. 248-258.

33. D.F.Clark and D.P.Casasent, "Practical optical Fourier analysis for high speed inspection", Opt. Eng. **27**, pp. 365-371 (1988).

34. C.Warde and J.Thackara, "Operating modes of the microchannel spatial light

modulator", Opt. Eng. **22**, pp. 695-699 (1983).

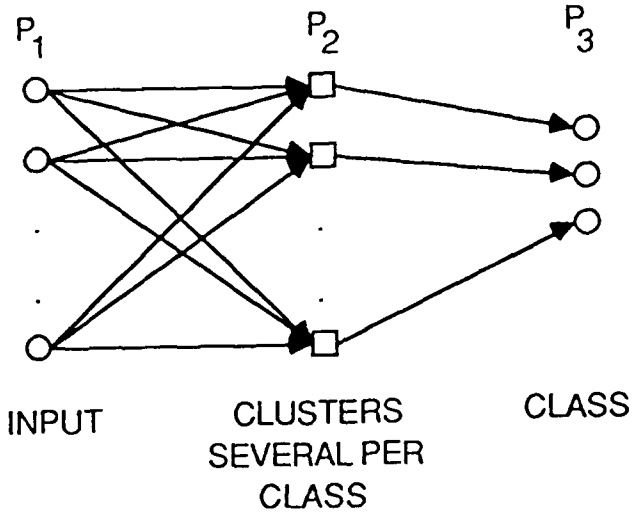


FIG. 1

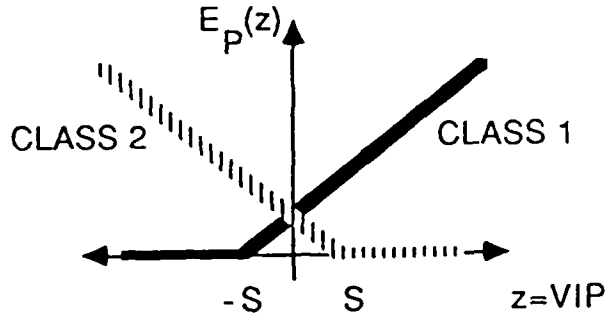


FIG. 2

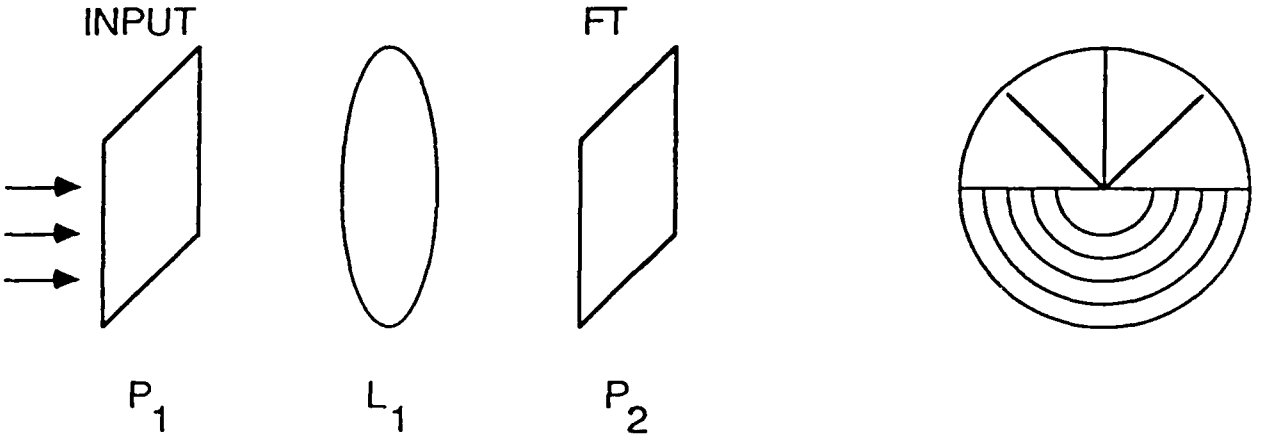


FIG. 3

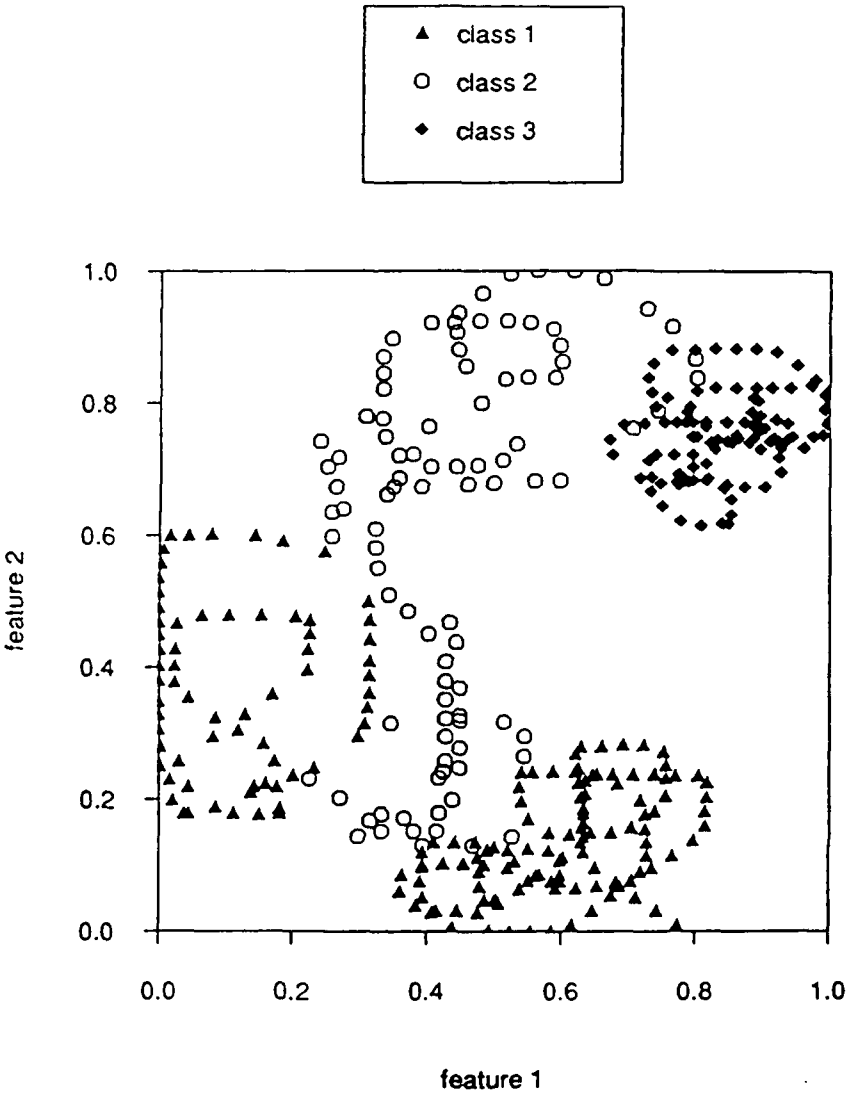

class 1
class 2
class 3
1.0
0.8
0.6
0.4
0.2
0.0
feature 2
0.0
0.2
0.4
0.6
0.8
1.0
feature 1

FIG. 4

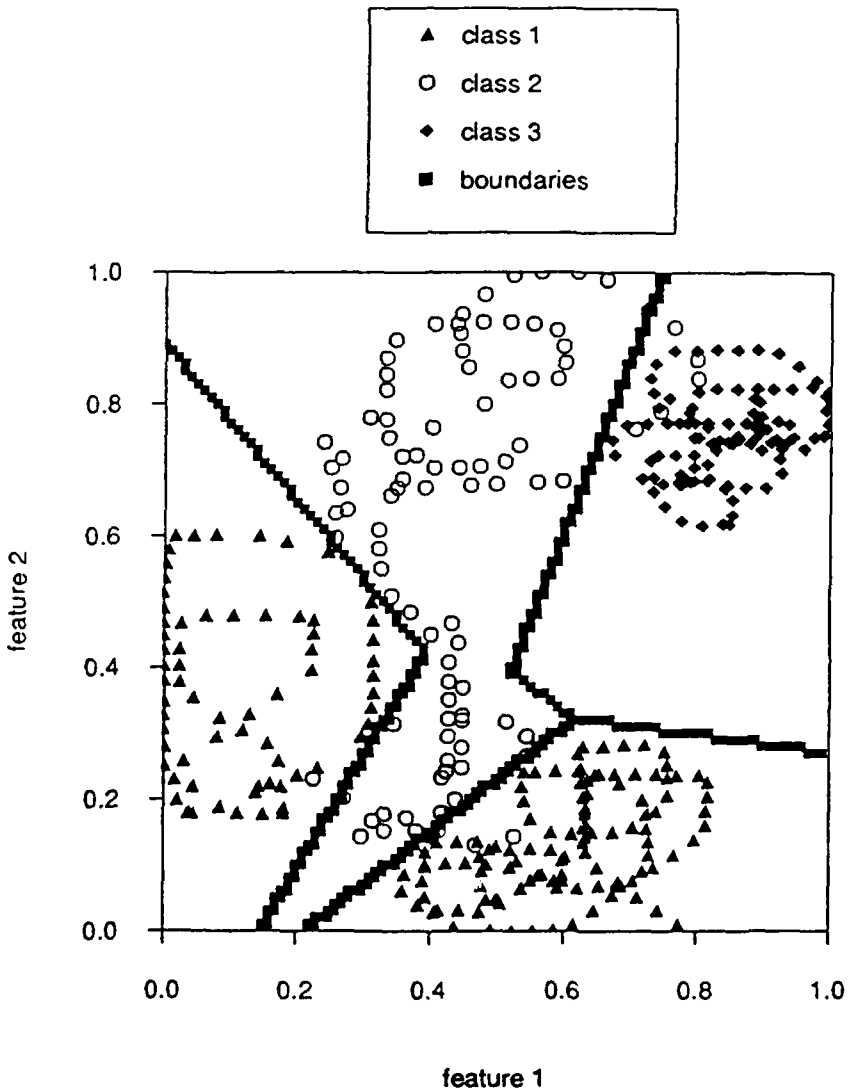

class 1
class 2
class 3
boundaries
1.0
0.8
0.6
0.4
0.2
0.0
feature 2
0.0
0.2
0.4
0.6
0.8
1.0
feature 1

Fig. 5

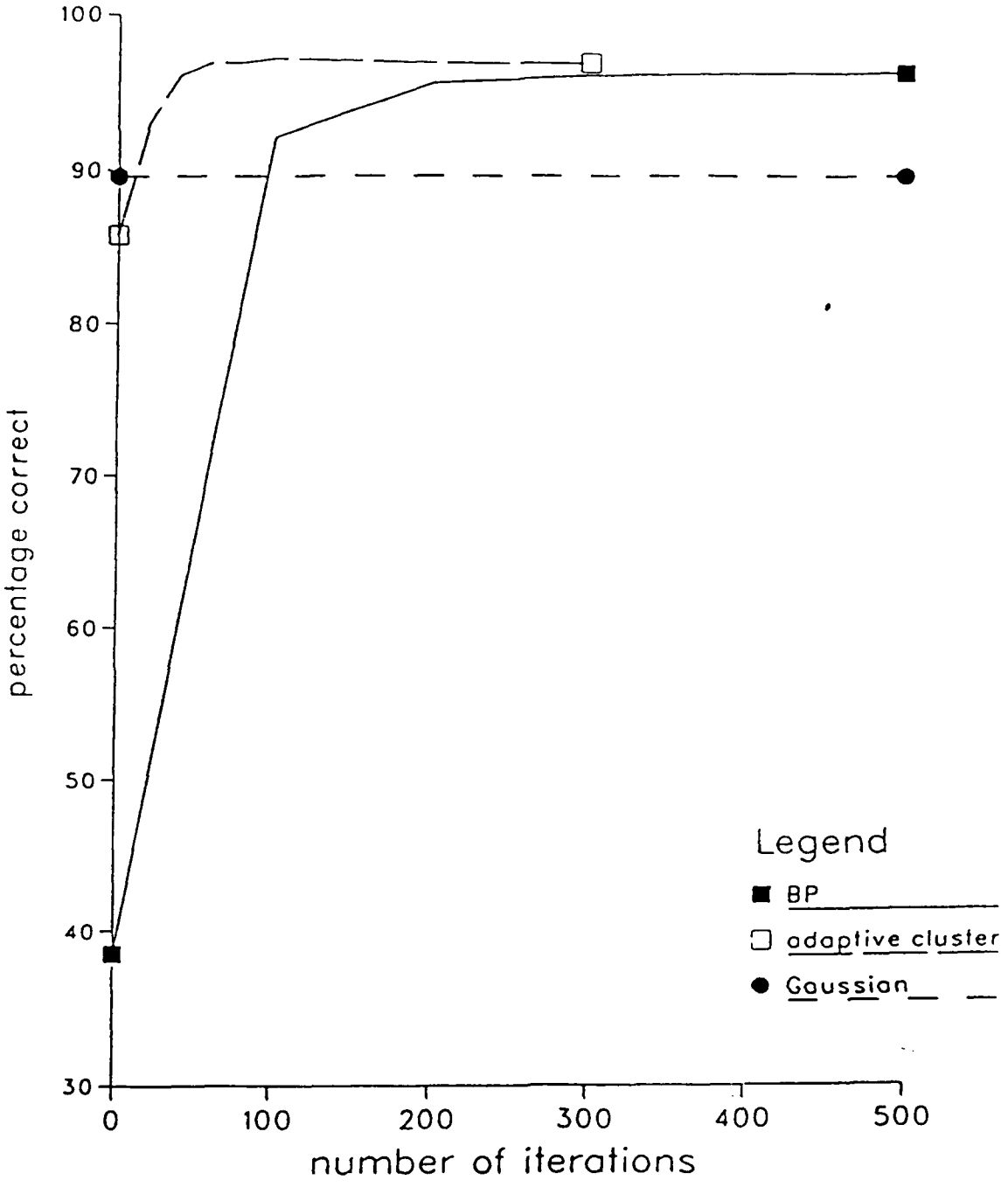

100
90
80
70
60
50
40
30
percentage correct
0
100
200
300
400
500
number of iterations
Legend
BP
adaptive cluster
Gaussian

FIG.6

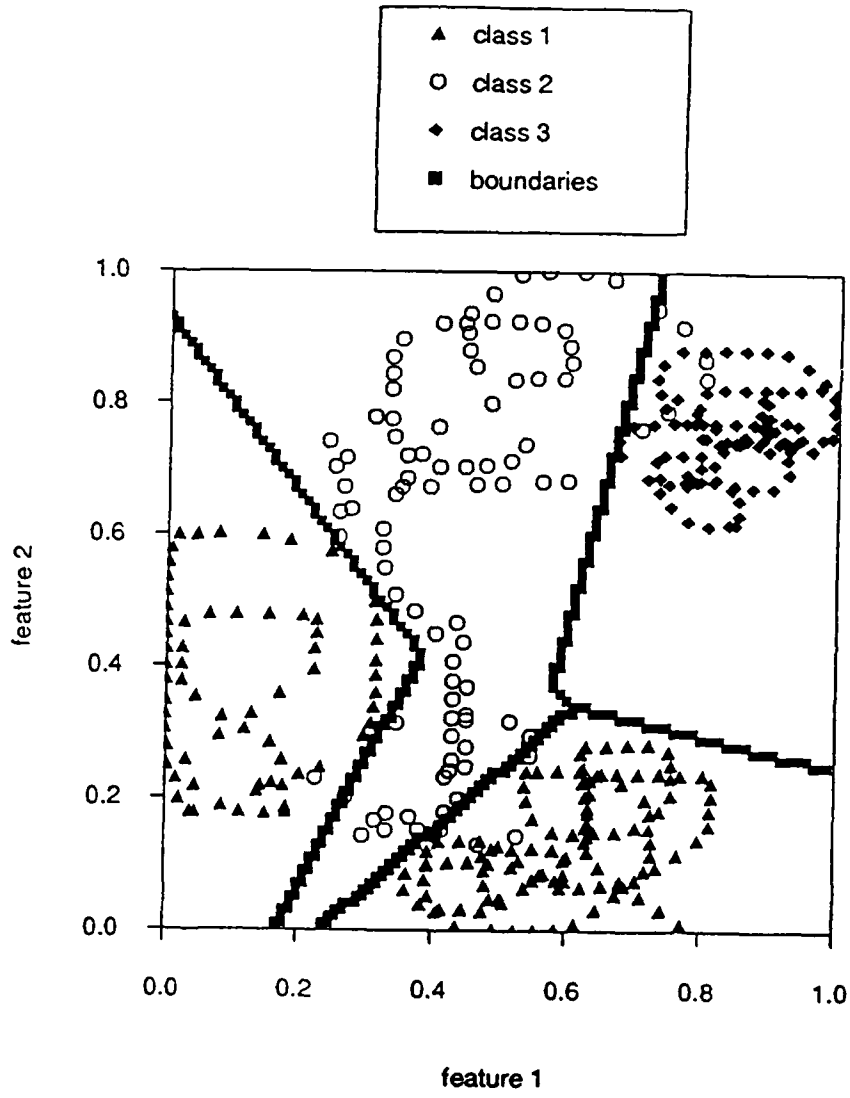

class 1
class 2
class 3
boundaries
1.0
0.8
0.6
0.4
0.2
0.0
feature 2
0.0
0.2
0.4
0.6
0.8
1.0
feature 1

FIG. 17

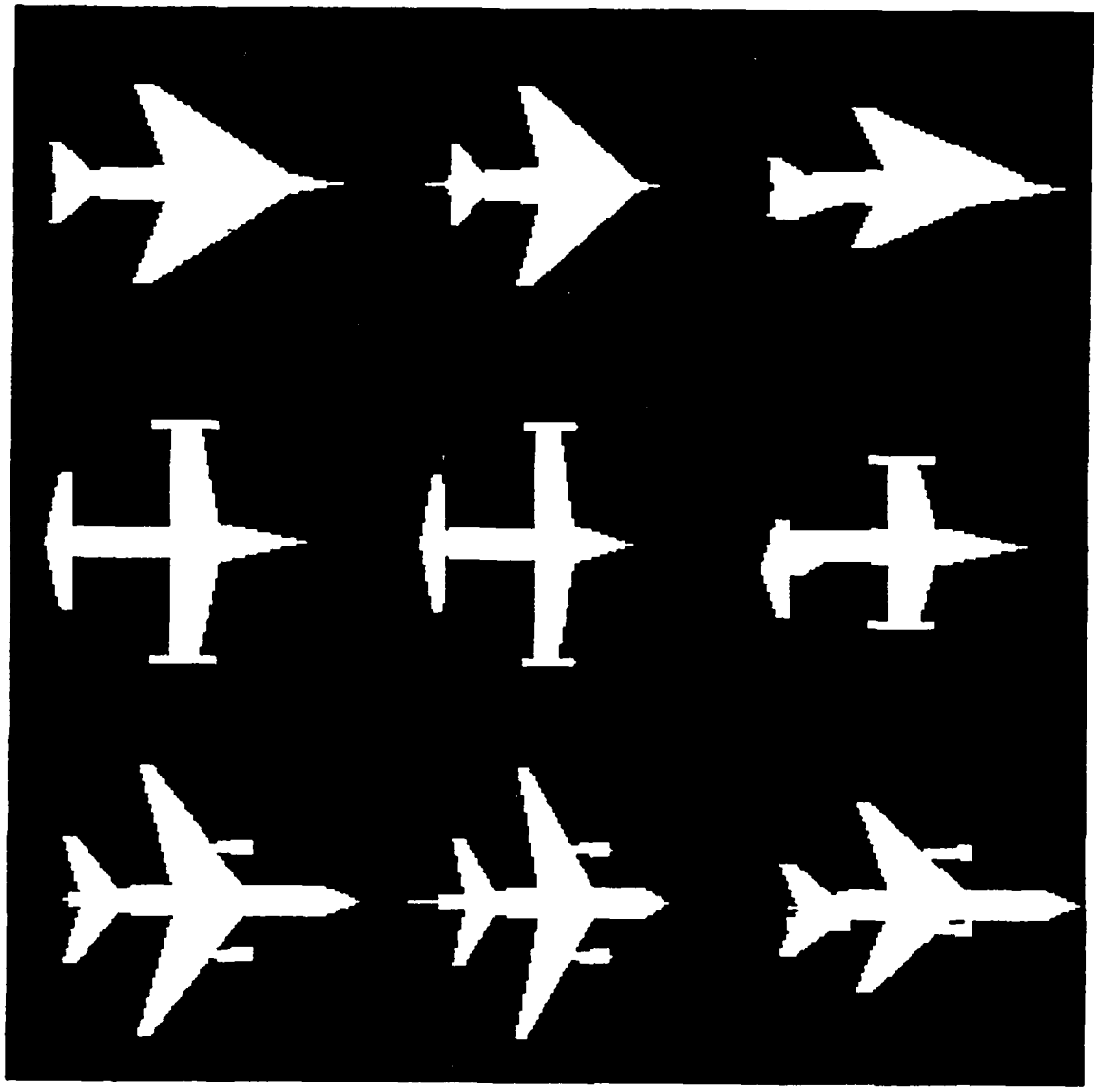


FIG. 8

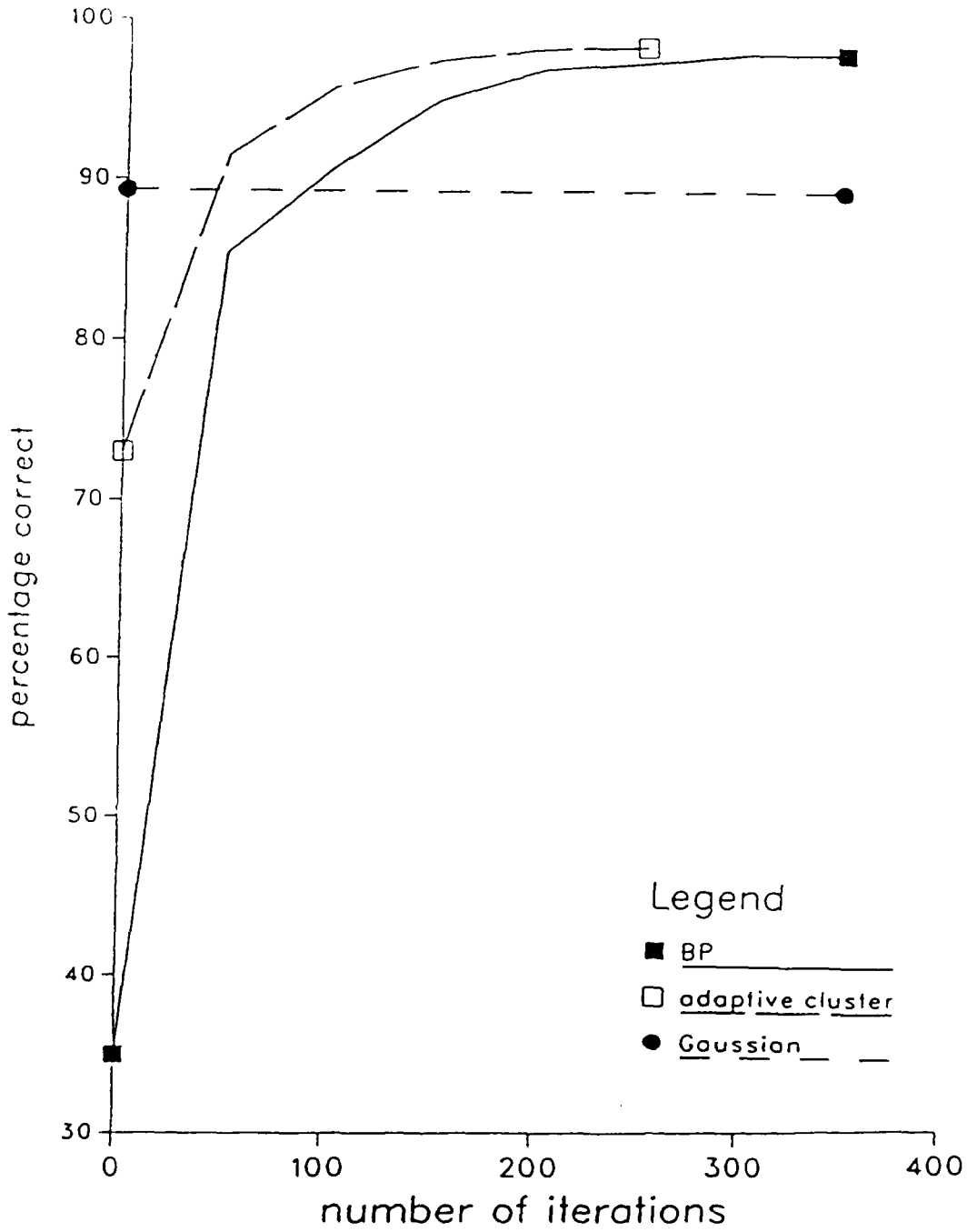

100
90
80
70
60
50
40
30
percentage correct
0
100
200
300
400
number of iterations
Legend
BP
adaptive cluster
Gaussian

FIG. 9

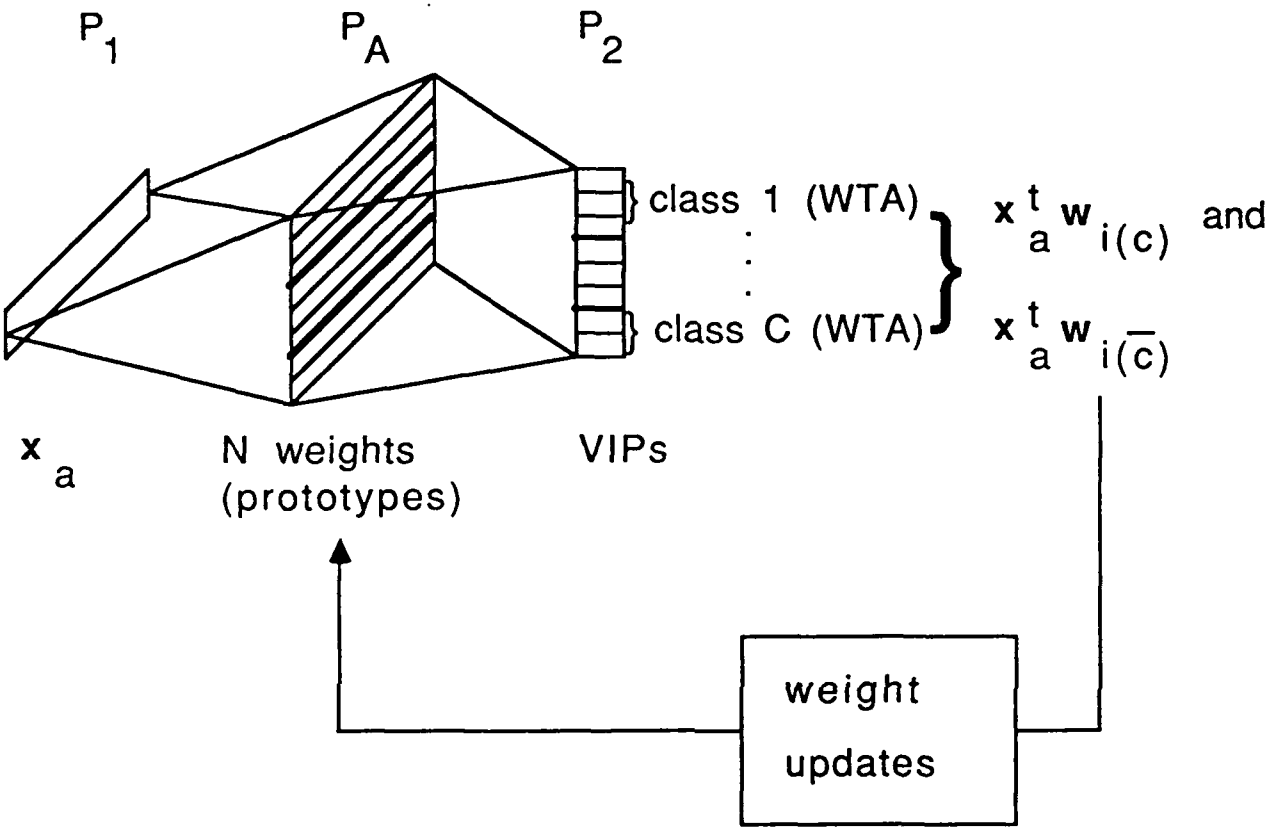
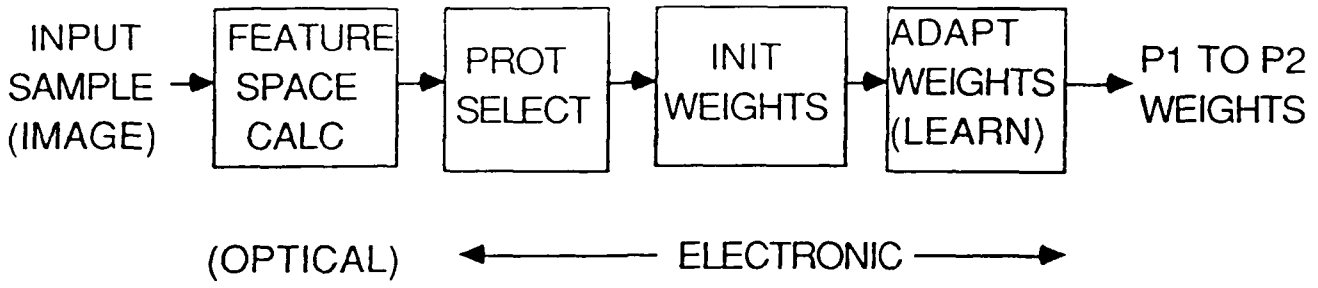
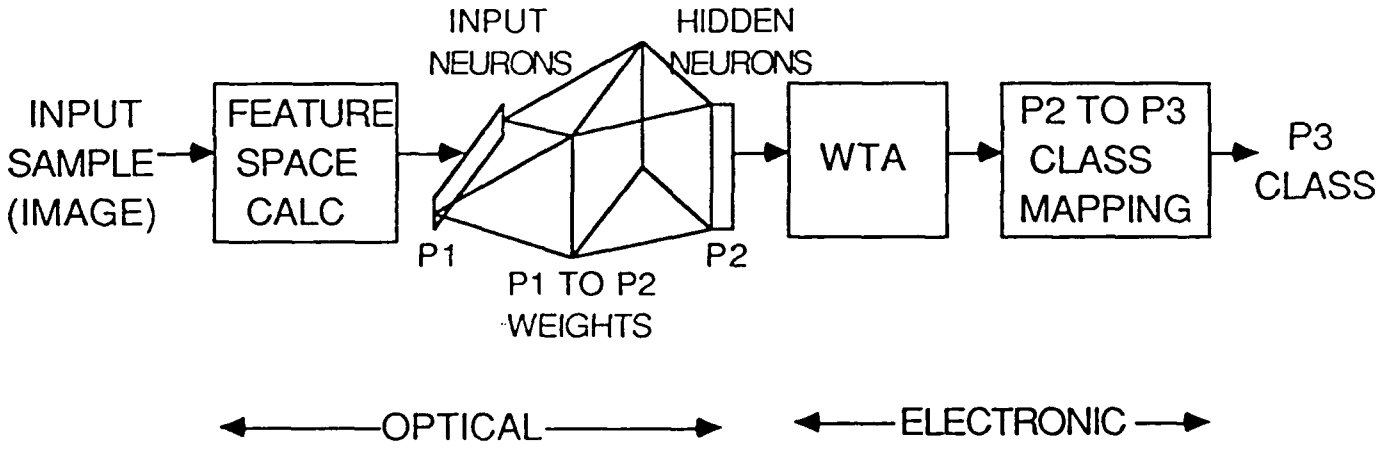

P1
PA
P2
class 1 (WTA)
class C (WTA)
x t a w i(c)
and
x t a w i(c̄)
x a
N weights
(prototypes)
VIPs
weight
updates

FIG. 10


INPUT SAMPLE (IMAGE)
FEATURE SPACE CALC
PROT SELECT
INIT WEIGHTS
ADAPT WEIGHTS (LEARN)
P1 TO P2 WEIGHTS
(OPTICAL)
ELECTRONIC

(a)


INPUT NEURONS
HIDDEN NEURONS
INPUT SAMPLE (IMAGE)
FEATURE SPACE CALC
P1
P2
P1 TO P2 WEIGHTS
WTA
P2 TO P3 CLASS MAPPING
P3 CLASS
OPTICAL
ELECTRONIC

(b)

FIG. 11